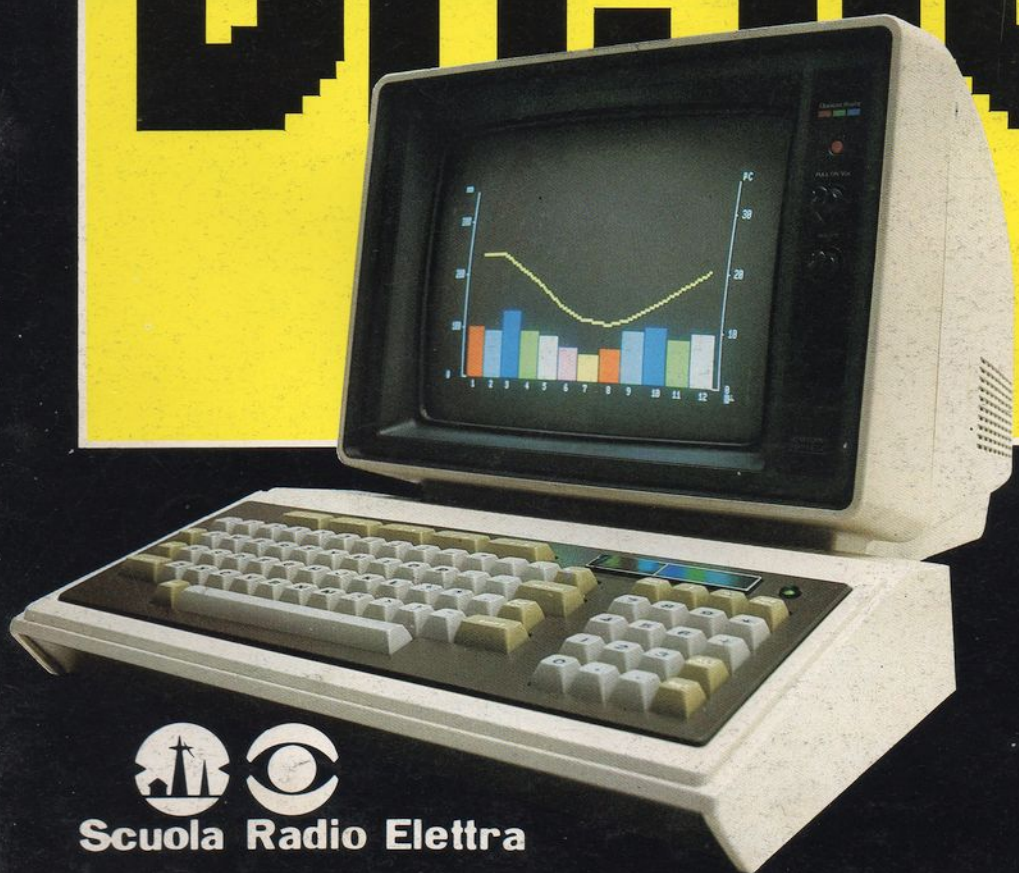
PARLA BASIC



Scuola Radio Elettra

Scuola Radio Elettra
Torino

Analisi e programmazione

Lezione 11

1 — Funzioni per il trattamento delle stringhe
2 — Funzioni complesse di stringa
3 — Altre funzioni BASIC

Che cosa apprenderà

In questa lezione verranno trattate le funzioni di stringa che consentono di operare logicamente su informazioni alfanumeriche, ossia su sequenze di lettere, numeri e simboli che vengono immessi nel computer tramite la tastiera.

Nel linguaggio BASIC sono previste otto funzioni stringa; esse vengono prevalentemente utilizzate per l'elaborazione di programmi di carattere gestionale-amministrativo.

Saranno infine analizzate altre funzioni BASIC che si usano nelle istruzioni di stampa per spaziare ed incolonnare dati.

1. - Funzioni per il trattamento delle stringhe

Nel linguaggio BASIC sono contemplate funzioni particolari che servono al trattamento di stringhe, le quali, come già è noto, sono raggruppamenti di uno o piú caratteri alfanumerici (termine già incontrato piú volte e che significa indifferentemente alfabetico o numerico).

Una stringa è un'entità di elaborazione la quale, al pari dei numeri, può essere manipolata, trasformata, operata con regole e funzioni appropriate, che nella sintassi del linguaggio BASIC prendono il nome di **funzioni di stringa** per analogia con le altre "funzioni matematiche" già esaminate nelle precedenti lezioni.

Una stringa può contenere al massimo 256 caratteri alfanumerici e va sempre assegnata a nomi di variabili stringa contenenti al massimo due caratteri alfanumerici (il primo dei quali è sempre una lettera), seguiti dal carattere speciale: $.

Per esempio, è corretta l'assegnazione:

```
S$= "SCUOLA"
```

ed è invece errata la seguente:

```
S="SCUOLA"
```

Infatti, come già precedentemente verificato, in quest'ultimo caso il computer emette il messaggio d'errore:

```
? TYPE MISMATCH ERROR
```

Il valore di ogni stringa è delimitato da due coppie di virgolette ("), per individuare l'inizio e la fine della stringa stessa.

Tra le virgolette di apertura e quelle di chiusura possono essere inseriti spazi, simboli grafici qualsiasi, caratteri d'interpunzione come punti e virgole, tranne naturalmente le virgolette.

Il numero di caratteri che possono essere contenuti in una variabile stringa varia da un computer all'altro.

Occorre tenere presente che anche gli spazi sono considerati caratteri, quindi la stringa:

```
A$= "MONTE BIANCO"
```

è formata da dodici caratteri.

Un valore numerico qualsiasi può essere rappresentato come stringa, racchiudendolo semplicemente tra virgolette. Ad esempio, l'istruzione:

```
1) P$="3,14159"
```

assegna il numero 3,14159 alla variabile stringa di nome P$, cosicché, se si volesse visualizzare questo numero sul video, si potrebbe scrivere, dopo avere eseguito l'istruzione 1), l'istruzione seguente:

```
2) PRINT P$
```

Eseguendo quest'ultima istruzione, sul video apparirà il valore 3,14159, cosí com'è assegnato alla variabile P$, **cioè proprio con la virgola dopo la prima cifra intera.**

Molto spesso in un programma si vogliono utilizzare, per scopi numerici, valori che però sono espressi come stringhe, oppure si vogliono visualizzare come stringhe valori che, all'interno della memoria, erano rappresentati come numeri.

Esistono due funzioni BASIC che servono a questo scopo, ossia che trasformano valori stringa in numeri e viceversa: la funzione **VAL** (X$) e la funzione **STR $**.

1.1 - Stringhe e numeri

La funzione **VAL** (X$) fa sí che la stringa di nome X$, contenente numeri, venga trasformata nel valore numerico da essa rappresentato.

Per esempio, se si scrivesse in modo immediato:

```
P1$="4"
P2$="-2.5"
PRINT VAL (P1$)*VAL (P2$)
```

sullo schermo dovrebbe apparire il valore numerico:

```
-10
```

risultato numerico corrispondente al prodotto dei due numeri 4 e -2.5, a cui sono state trasformate le "stringhe" numeriche:

```
P1$="4"
P2$="-2.5"
```

Se invece si fosse scritto:

```
P2$="-2,5" anziché P2$="-2.5"
```

si sarebbe ottenuto il valore -8, perché la funzione **VAL** (P2$) avrebbe fornito come risultato il valore -2 (si tenga presente che la funzione **VAL** "cerca" d'interpretare, a partire da sinistra, come numero il valore -2,5, ma ci riesce solo a metà, perché la virgola non è contemplata nella sintassi BASIC per la rappresentazione numerica).

Come constatato dagli esempi ora fatti, la funzione **VAL** (X$), abbreviazione della parola inglese "value", ossia valore, permette di ricavare i valori numerici contenuti in una stringa, i quali sarebbero trattati come qualsiasi altro carattere alfanumerico.

L'argomento della funzione **VAL** è costituito unicamente da stringhe che iniziano con cifre decimali. Altri numeri contenuti nella stessa stringa, separati dai primi da caratteri alfabetici o simboli, non vengono presi in considerazione.

Gli esempi che seguono chiariranno ulteriormente l'operatività della funzione **VAL**.

Se si scrivono le istruzioni immediate:

```
A$="25 DICEMBRE 1985"
PRINT VAL (A$)
```

sullo schermo comparirà:

```
25
```

perché il numero 1985, essendo posto dopo la parola DICEMBRE, non viene considerato.

Se si scrive:

```
B$="ANNO 1985"
PRINT VAL (B$)
```

sullo schermo apparirà:

```
0
```

perché la stringa non inizia con un numero decimale.

Scrivendo:

```
C$="23*25"
PRINT VAL (C$)
```

sullo schermo apparirà solamente il numero 23, perché dopo tale numero è presente il carattere *.

Funzione STR$ (X)

La funzione **STR$** (X) agisce in modo opposto all'istruzione **VAL**: essa cioè fa sí che un valore numerico X venga interpretato e trasformato come valore stringa.

Ad esempio, l'istruzione:

```
PRINT STR$ (3.14159)
```

fa apparire sullo schermo il valore:

```
3.14159
```

così come è scritto prima.

In pratica, viene visualizzato il valore stringa corrispondente al valore numerico X, argomento della funzione **STR$** (X).

1.2 - Variabili stringa e codice ASCII

Il linguaggio BASIC dispone di due funzioni, denominate **ASC** (X$) e **CHR$** (X), le quali consentono rispettivamente di ottenere il codice ASCII di un carattere e, viceversa, di fornire il carattere dato il relativo codice.

Come già precisato nella lezione *Analisi e programmazione 2,* mediante il codice ASCII tutti i simboli riportati sulla tastiera sono codificati in caratteri binari nel momento in cui i tasti vengono pigiati.

Tale codificazione è indispensabile perché il computer, nel suo interno, può elaborare soltanto numeri in forma binaria.

Verifichiamo ora l'operatività delle due funzioni in questione.

Funzione ASC (X$)

La funzione **ASC** (X$) fornisce il valore numerico intero (del sistema decimale) corrispondente al codice binario ASCII del primo carattere della variabile stringa X$.

Se, per esempio, X$ contiene la lettera "A" dell'alfabeto, eseguendo l'istruzione:

```
X$="A"
PRINT ASC (X$)
```

verrà visualizzato il numero intero 65, che è appunto il valore decimale corrispondente al valore binario nel codice ASCII della lettera A (ved. tabella della *fig. 3* della lezione *Analisi e programmazione 2*).

Analogamente, se a X$ viene assegnata la stringa "SCUOLA", scrivendo ad esempio:

```
PRINT ASC (X$)
```

verrà visualizzato il numero intero 83, che è il codice decimale corrispondente al codice binario ASCII della lettera S, posta in prima posizione nella parola "SCUOLA".

Funzione CHR$ (X)

La funzione **CHR$** (X) agisce in modo opposto alla funzione **ASC**, ossia fornisce il carattere stringa corrispondente al valore intero X che, a sua volta, è la rappresentazione decimale di un certo codice binario ASCII.

Per esempio, se si scrive la seguente istruzione BASIC:

```
PRINT CHR$(66)
```

si vedrà apparire sullo schermo il carattere letterale B.

Oppure, se si scrive:

```
X=37
PRINT CHR$(X)
```

apparirà il carattere %.

1.3 - Operazioni con le stringhe

Le funzioni **LEN**, **MID$**, **RIGHT$** e **LEFT$** servono per "operare" sulle parole e sono molto utilizzate nel cosiddetto "word processing" (termine inglese, usato per indicare l'elaborazione di testi letterali).

Funzione LEN (A$)

La funzione **LEN** (X$) fornisce la lunghezza (in carattere) del valore stringa contenuto nella variabile di nome X$. La parola chiave LEN, infatti, è l'abbreviazione del termine inglese "length", che vuol dire appunto lunghezza.

Se si esegue ad esempio l'istruzione BASIC ad esecuzione immediata

```
PRINT LEN ("PESCASSEROLI")
```

sullo schermo TV apparirà il numero intero

```
12
```

che rappresenta il numero di caratteri della parola "PESCASSEROLI".

Le virgolette di apertura e chiusura di una stringa, come già precisato, non fanno parte del conteggio dei caratteri; esse infatti rappresentano dei "delimitatori" per le stringhe.

Ecco un altro esempio: se si esegue l'istruzione BASIC

```
PRINT LEN ("3,14159")
```

sullo schermo apparirà il numero 7, corrispondente alla quantità dei caratteri del numero

3,14159

rappresentato come stringa.

Come si è potuto notare, dentro la parentesi della funzione LEN c'è sempre il valore o il nome di una variabile stringa.

Un ultimo esempio: se si scrive l'istruzione

```
S1$="SCUOLA RADIO ELETTRA"
PRINT LEN (S1$)
```

si vedrà apparire sullo schermo il numero intero

20

cioè il numero dei caratteri (compresi gli spazi interni) dell'espressione letterale:

SCUOLA RADIO ELETTRA.

Funzione MID$

La funzione stringa **MID$** (X$, S, X) estrae il numero X di caratteri dalla stringa di nome X$, a partire dalla posizione (carattere) S della stessa stringa.

Per esempio, se si scrive l'istruzione BASIC ad esecuzione immediata

```
PRINT MID$ ("ROMA",3,2)
```

specifica il carattere da cui partire

numero di caratteri da estrarre

sullo schermo apparirà la "semiparola"

MA

che è la parte finale della stringa "ROMA". Come constatato, la funzione **MID$** ha estratto, in questo caso, due caratteri (MA) a partire dal terzo carattere (M) della stringa considerata.

Con il seguente, semplice programma BASIC dovremmo ora vedere sullo schermo una specie di anagramma avente l'aspetto sotto riportato.

R
RO
ROM
ROMA

Il listato del programma che fornisce tale anagramma è indicato di seguito.

```
5 PRINT"□"
10 PA$="ROMA"
15 LU=LEN(PA$)
20 FOR X=1 TO LU
25 PRINT TAB(8);MID$(PA$,1,X)
30 NEXT X
35 STOP
40 END
```

L'istruzione 25, compresa nel ciclo **FOR... NEXT**, visualizzerà sullo schermo la prima volta la lettera R, che occupa la prima posizione nella parola "ROMA" ed a cui è stato assegnato il nome PA$; la seconda volta visualizzerà il gruppo di caratteri RO, che costituiscono le prime due lettere della stessa stringa; la terza volta visualizzerà il gruppo ROM, prime tre lettere di ROMA, e la quarta volta l'intera parola "ROMA".

Con le funzioni stringa **LEN** e **MID$** si possono effettuare svariati artifici di stampa e composizioni grafiche molto utilizzate sia nei video-giochi sia nella elaborazione di moduli e prospetti grafici.

Funzione LEFT$ (X$,X)

La funzione **LEFT$** (X$,X) fornisce una stringa formata dai primi X caratteri posti piú a sinistra ("left" in inglese significa sinistra) della stringa X$ indicata come argomento.

Ad esempio, se si scrive l'istruzione BASIC ad esecuzione immediata

PRINT LEFT$ ("MELOGRANO",4)

verrà visualizzata sullo schermo la stringa

MELO,

costituita appunto dai primi quattro caratteri della stringa originale "MELOGRANO".

Scrivendo

PRINT LEFT$ ("LEONARDO DA VINCI",8)

comparirà sullo schermo la stringa

LEONARDO.

Nel caso in cui il numero X di caratteri da estrarre sia superiore alla lunghezza della stringa considerata, viene visualizzata sullo schermo la stringa completa.

Per esempio, se si scrive

PRINT LEFT$ ("ITALIA",7)

sullo schermo si vedrà la stringa

ITALIA

Funzione **RIGHT$** (X$,X)

La funzione **RIGHT$** (X$,X) fornisce invece una stringa formata dai primi X caratteri posti piú a destra ("right" in inglese significa appunto destra) della stringa X$ indicata come argomento.

Per esempio, se si scrive

PRINT RIGHT$ ("MELOGRANO",5)

si vedranno apparire i cinque caratteri piú a destra della stessa stringa, cioè

GRANO.

Anche in questo caso, se il numero X di caratteri da estrarre è maggiore

della quantità di caratteri costituenti da stringa considerata, sullo schermo viene visualizzata la stringa completa.

I risvolti applicativi delle tre funzioni stringa ora esaminate sono riassunti nell'esempio riportato nella *fig. 1*, nella quale è mostrato come la stringa "SCUOLA RADIO ELETTRA" possa essere scomposta, ad opera delle funzioni **LEFT$**, **MID$** e **RIGHT$**, in tre stringhe (sottostringhe).

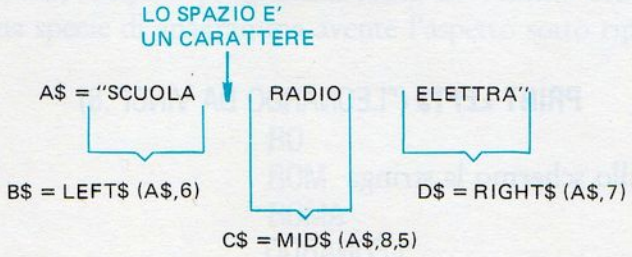


Fig. 1 - Esempio d'impiego delle tre funzioni di stringa **LEFT$**, **MID$** *e* **RIGHT$**.

Somma di stringhe

Le stringhe alfanumeriche si possono vantaggiosamente combinare fra loro per formare stringhe piú grandi.

A questo fine viene utilizzato il segno operativo + della somma, che appunto aggiunge ad una stringa i caratteri di un'altra stringa.

In questo modo si possono combinare, scomporre e ricomporre intere parole utilizzando le funzioni stringa.

Per esempio, si può eseguire con istruzioni immediate la somma delle due stringhe che seguono:

```
PA$="AUTO"
PB$="STRADA"
PRINT PA$+PB$
```

Sullo schermo comparirà la parola

AUTOSTRADA

Scrivendo:

```
PA$="MELOGRANO"
PRINT LEFT$ (PA$, 4)+RIGHT$ (PA$,5)
```

sullo schermo si vedrà la parola

MELOGRANO

Quale ulteriore esempio, si può scrivere:

```
P1$="SCUOLA"
P2$="RADIO"
P3$="ELETTRA"
SP$="  "          2 spazi
PRINT P1$+SP$+P2$+SP$+P3$
```

nel qual caso si vedrà comparire sullo schermo l'espressione:

SCUOLA RADIO ELETTRA.
(2 spazi, 2 spazi)

Come si vede, dopo la prima parola (SCUOLA) e prima dell'ultima parola (ELETTRA) si sono interposti due spazi (assegnati alla stringa SP$).

2. - Funzioni complesse di stringa

Le funzioni considerate prima si possono definire funzioni semplici di stringa, perché elaborano in modo semplice appunto i caratteri alfanumerici.

Analogamente a quanto visto nelle precedenti lezioni per i calcoli di funzioni matematiche, esiste anche per le stringhe la possibilità di effettuare operazioni piú complesse ed a questo scopo si ricorre alle cosiddette **funzioni complesse di stringa**, per il cui utilizzo si segue la stessa prassi adottata per il calcolo di funzioni matematiche.

Si definisce cioè prima, generalmente nelle prime posizioni di un programma, una funzione o stringa con l'istruzione di definizione **DEF FN**, seguita dal nome di una variabile stringa che funge da risultato di un certo calcolo (il nome della stringa va scritto secondo le normali regole per la denominazione delle variabili).

A destra del nome si riporta tra parentesi il nome della variabile stringa indipendente.

Impiego delle funzioni stringa

Le funzioni stringa sono spesso utilizzate per separare una o piú parti numeriche contenute in un campo alfanumerico.

Per esempio, si consideri il codice AØ1PØ3 adottato da un'azienda per contraddistinguere la pressa PØ3, situata nel reparto Ø1 dello stabilimento A.

In questo caso numeri e lettere servono ad identificare, nell'ambito dei reparti produttivi, un particolare macchinario.

Normalmente le lettere vengono utilizzate per specificare collocazioni e tipi di prodotti, mentre i numeri servono ad individuarne la quantità.

Nel caso considerato si tratterebbe quindi della pressa n. 3, situata nel reparto Ø1 dello stabilimento A.

Un programma di elaborazione che volesse stabilire quante e quali presse ci sono nel reparto Ø1 dello stabilimento A dovrebbe "indagare" sui codici d'archivio-macchine, aventi i primi quattro caratteri alfanumerici uguali alla parte costante

PARTE COSTANTE

A01PXX

e contare quanti valori XX ci sono in questi codici.

Se si assegna un codice generico al nome di variabile stringa C$, si può scrivere in linguaggio BASIC

```
N=VAL (MID$ (C$),5,2).
```

In questo modo la variabile numerica N assumerà proprio il valore XX che si desidera conoscere.

Esso è infatti il valore numerico (VAL) degli ultimi due caratteri della stringa AØ1PXX (di nome C$), cioè dei caratteri che partono dalla quinta posizione della stringa stessa, come specificato qui sotto.

caratteri stringa

1°	2°	3°	4°	5°	6°
A	Ø	1	P	X	X

Come si vede da questo semplice esempio, le funzioni stringa consentono di eseguire numerose operazioni sulle stringhe di caratteri.

Esse effettuano il cosiddetto **trattamento logico** delle parole, in analogia con le funzioni matematiche che effettuano il "trattamento aritmetico" dei numeri.

Inoltre, se si considera che una stringa può contenere anche caratteri di controllo, codici di colori e simboli grafici, note musicali, codici di rumore, ecc., risulta evidente che è possibile programmare sul computer, con l'ausilio delle funzioni stringa, una vasta gamma di figure geometriche statiche o in movimento, a colori o in b/n, e simultaneamente produrre effetti sonori e motivi musicali.

Tutto ciò naturalmente va inserito in un'elaborazione, senza alterare lo scopo principale di un programma che è quasi sempre di natura digitale o numerica.

Gli effetti grafici devono essere quindi utilizzati come ausilio di un'elaborazione sul computer, il quale, non va dimenticato, è uno strumento adatto al trattamento di informazioni di natura essenzialmente digitale.

Per esempio, scrivendo

```
DEF FN A$ (X$)=LEFT $ (X$,3)+RIGHT$ (X$,3)
```

viene definita una funzione stringa chiamata A$, dipendente dalla variabile stringa X$ (quest'ultima, ad esempio, contenente un codice alfanumerico).

Se quindi nel prosieguo del programma BASIC si ha necessità di disporre della combinazione stringa dei primi tre e degli ultimi tre caratteri di un certo codice alfanumerico (per esempio, il codice "1Ø1ABC2ØØ"), si potrà scrivere

```
CØ$=FN A$("1Ø1ABC2ØØ")
```

per assegnare la stringa "1Ø12ØØ" alla variabile stringa di nome CØ$.

Come sempre, è il sistema BASIC che provvede a calcolare la funzione complessa di stringa definita precedentemente, la quale consisteva nel "prelevare" da una qualsiasi stringa alfanumerica (nel nostro caso, un codice) i primi tre e gli ultimi tre caratteri, cioè i numeri 1Ø1 e 2ØØ.

Non tutti i linguaggi BASIC dispongono delle funzioni complesse di stringa. In loro mancanza, quando si devono effettuare operazioni speciali sulle stringhe, si ricorre alle subroutine che, come si è visto, hanno gli stessi effetti pratici delle funzioni.

3. - Altre funzioni BASIC

Oltre che le funzioni matematiche e per il trattamento delle stringhe, il linguaggio BASIC dispone di funzioni che si usano nelle istruzioni di stampa per spaziare i dati fra loro e per incolonnarli come in una comune macchina per scrivere.

Funzione TAB (X)

La piú importante di queste funzioni è la **TAB** (X), già trattata nelle precedenti lezioni, la quale consente di "tabulare" un numero o una stringa nella posizione di colonna indicata dal valore di X.

Tale funzione si utilizza in un'istruzione **PRINT** per visualizzare i valori (numerici o stringa, appunto) in determinati punti dello schermo.

Per esempio, se si scrive la seguente istruzione ad esecuzione immediata

```
PRINT TAB (11);"A"
```

sullo schermo apparirà la lettera A in corrispondenza della colonna 11 (dodicesima colonna) dello schermo stesso.

Il numero X parte da Ø e può arrivare al valore massimo consentito dal particolare tipo di personal computer che si impiega.

Per esempio, se si utilizza il VIC-20 (che ha lo schermo di 22 colonne, numerate da Ø a 21) l'istruzione

PRINT TAB (Ø);"A";**TAB** (21);"S"

farà apparire la lettera A in corrispondenza del margine sinistro dello schermo (colonna Ø) e la lettera S in corrispondenza del margine destro (colonna 21).

Scrivendo invece

PRINT TAB (22);"D"

apparirà sullo schermo, in corrispondenza della colonna Ø, la lettera D.

Questo perché si è chiesto di stampare la lettera D su una colonna con numero superiore a quello massimo (21) previsto dal computer, per cui la lettera stessa viene posizionata in corrispondenza della colonna Ø.

Utilizzando invece un computer di maggiori prestazioni, con schermo di 4Ø colonne, l'esecuzione dell'istruzione

PRINT TAB (Ø);"A";**TAB** (1Ø);"S";**TAB** (2Ø);"D";**TAB** (3Ø);"F"

farà apparire su una linea dello schermo le lettere A S D F in corrispondenza rispettivamente delle colonne Ø, 1Ø, 2Ø, 3Ø che contraddistinguono l'inizio delle quattro zone di stampa del computer.

La funzione **SPC** (X) invece viene usata con l'istruzione **PRINT** quando si vuole spaziare fra loro due dati o due stringhe.

Per esempio, se si desidera visualizzare l'espressione

CASA ␣ BELLA

5 spazi

con cinque spazi di separazione fra le due parole, si può scrivere la seguente istruzione BASIC ad esecuzione immediata

```
PRINT "CASA"; SPC (5);"BELLA"
```

spazi di separazione

Volendo separare i due numeri

32 26

8 spazi

con otto spazi, si scriverà

```
PRINT 32; SPC (8); 26
```

La funzione **POS** (X) fornisce la posizione (o numero di colonna dello schermo) in cui si trova il cursore; X può avere qualsiasi valore.

Funzione POS (X)

La funzione **POS** deve essere usata con l'istruzione **PRINT**.

Per esempio, se si scrive:

```
10 PRINT POS (3)
20 PRINT "COMPUTER"
```

e si esegue il programma con il comando **RUN**, sullo schermo apparirà il numero Ø, relativo alla colonna da cui parte la scritta COMPUTER.

Dopo avere cancellato la memoria del computer ed eseguito le istruzioni che seguono:

```
10 PRINT "ROSA";
20 PRINT POS (3);
30 PRINT "BIANCA"
```

sullo schermo dovrebbe apparire la scritta (a partire dal margine sinistro):

ROSA 4 BIANCA

1 spazio — 1 spazio

Il numero 4 scritto fra le due parole indica la colonna (la quinta) dalla quale dovrebbe iniziare la successiva parola (BIANCA).

Funzione FRE (X)

La funzione **FRE (X)**, infine, fornisce il numero dei byte liberi, ancora disponibili nella memoria principale del computer ed utilizzabili per la programmazione indipendentemente dal valore di X.

Pertanto, il numero X posto fra parentesi può avere qualsiasi valore.

Questa funzione è molto utile per conoscere "a priori" quanta memoria si ha ancora a disposizione per aggiungere altre istruzioni in un programma.

Occorre tenere presente che la memoria libera per l'utente è tutto quanto resta dalla differenza fra la massima capacità (misurata in Kbyte) e quella occupata dalle istruzioni di programma BASIC.

La funzione **FRE (X)** deve essere sempre usata con l'istruzione **PRINT** ossia:

PRINT FRE (1)

In alcuni tipi di computer, come per esempio nel C64, il valore dei byte disponibili, ottenuto con la funzione **FRE**, è espresso con un numero negativo.

In tal caso, per potere conoscere il numero effettivo dei byte disponibili occorre eseguire un semplice calcolo.

In primo luogo si determina il numero complessivo dei byte relativi alla massima capacità di memoria del computer, tenendo presente che 1 Kbyte = 1024 byte.

Pertanto, nel caso del C64, facendo il prodotto $64 \times 1024 = 65536$ byte, si ottiene la capacità della memoria totale di cui il computer dispone all'atto della sua accensione.

Sottraendo infine dal numero ottenuto, 65536 nel caso ipotizzato prima, il valore numerico negativo visualizzato sullo schermo con la funzione **FRE (X)**, si conosce il dato desiderato, ossia la quantità di memoria ancora disponibile in quel momento ed utilizzabile per i programmi.

* * * * * * *

In seguito...

Nella prossima lezione verrà descritto il modo in cui il computer gestisce le informazioni.

A questo scopo si esamineranno le organizzazioni dei dati nella memoria principale del calcolatore e nelle memorie ausiliarie (unità a nastro o a disco) ed i comandi in linguaggio BASIC per la lettura e la scrittura dei dati stessi.

Da memorizzare

- Una stringa è **un'entità di elaborazione** che, al pari dei numeri, può essere manipolata, trasformata, operata con regole e funzioni appropriate, che prendono il nome di **funzioni di stringa**.
- La funzione **VAL$** converte una stringa contenente numeri nel valore numerico da essa rappresentato.
- La funzione **STR$** trasforma in stringa un valore numerico.
- La funzione **ASC** (X$) fornisce il valore numerico decimale corrispondente al codice binario ASCII del primo carattere della variabile stringa.
- La funzione **CHR$** (X) fornisce il carattere stringa corrispondente al valore di X relativo ad un determinato codice binario ASCII.
- La funzione **LEN** (X$) fornisce la lunghezza in caratteri di una stringa X$.
- La funzione **MID$** (X$, S, X) estrae il numero X di caratteri dalla stringa di nome X$, a partire dalla posizione S della stringa stessa.
- La funzione **LEFT$** (X$, X) fornisce una stringa formata dai primi X caratteri posti piú a sinistra della stringa X$.
- La funzione **RIGHT$** (X$, X) fornisce una stringa formata dai primi X caratteri posti piú a destra della stringa X$.
- Le stringhe si possono sommare tra loro utilizzando il segno operativo + della somma.
- La funzione **TAB** (X) consente di specificare il numero della colonna in cui si desidera sia visualizzato il prossimo carattere.
- La funzione **SPC** (X) permette di separare fra loro due dati o due stringhe di X spazi.
- La funzione **POS** (X) fornisce il numero della colonna in corrispondenza della quale si trova il cursore dello schermo.
- La funzione **FRE** (X) permette di conoscere il numero dei byte liberi, ancora disponibili nella memoria principale del computer ed utilizzabili per la programmazione.

Scuola Radio Elettra
Torino

Pratica del BASIC

Lezione 11

1 — Il codice ASCII dei computer C64 e VIC-20
2 — Esercitazioni con le stringhe
3 — Le funzioni di tempo
4 — Animazione dello sprite con il C64
5 — Suoni ed effetti sonori con il C64
6 — Suoni ed effetti sonori con il VIC-20

Che cosa farà

Nella prima parte della lezione si esamineranno i codici ASCII del Suo computer, quindi potrà verificare l'operatività delle funzioni stringa utilizzate nel linguaggio BASIC, eseguendo appositi esercizi di programmazione.

Nella seconda parte della lezione si descriverà il sistema per animare gli sprite sul C64 e si completerà l'esame delle possibilità offerte dai computer C64 e VIC-20 nel campo musicale.

1. – Il codice ASCII dei computer C64 e VIC-20

Come già precedentemente precisato, i simboli e i caratteri riportati sulla tastiera sono codificati, nell'interno del computer, mediante codici binari (Sistema ASCII).

Occorre subito far notare che la serie di caratteri ASCII non è uguale per tutti i computer, anche se la maggior parte di essi utilizza, seppur parzialmente, tale sistema di codifica.

I codici binari ASCII dei microcomputer Commodore (C64 e VIC-20) sono 192 (numerati da Ø a 191); quasi tutti questi codici rappresentano i caratteri riportati sulla tastiera (caratteri letterali, cifre decimali, caratteri grafici, di controllo del cursore, ecc.); i rimanenti codici sono inutilizzati.

Nelle tabelle della *fig. 1* e della *fig. 2* sono riportati i caratteri della tastiera del C64 e del VIC-20 (prima e terza colonna) ed i corrispondenti codici ASCII espressi in notazione decimale (seconda e quarta colonna).

Osservando tali tabelle possiamo fare le seguenti considerazioni.

- I primi 32 codici sono in parte relativi ai tasti per il controllo dello schermo e di alcuni colori. Nel computer VIC-20 i codici 8 e 9 sono liberi.

- I codici compresi fra 33 e 90 corrispondono a caratteri alfanumerici, agli operatori aritmetici, agli operatori relazionali, alle dieci cifre decimali ed alle lettere maiuscole dell'alfabeto. I codici da 65 a 90 sono praticamente uguali per tutti i tipi di computer.

- I codici da 91 a 95 sono relativi ad altri simboli alfanumerici.

- I codici da 96 a 127 corrispondono ai caratteri grafici residenti nella memoria ROM del computer e vengono attivati premendo contemporaneamente il tasto **SHIFT** .

CARATTERI	CODICE ASCII	CARATTERI	CODICE ASCII
	0	0	48
	1	1	49
	2	2	50
	3	3	51
	4	4	52
WHT	5	5	53
	6	6	54
	7	7	55
DISABILITA SHIFT C=	8	8	56
ABILITA SHIFT C=	9	9	57
	10	:	58
	11	;	59
	12	<	60
RETURN	13	=	61
SWITCH TO LOWER CASE	14	>	62
	15	?	63
	16	@	64
CRSR ↓	17	A	65
RVS ON	18	B	66
CLR HOME	19	C	67
INST DEL	20	D	68
	21	E	69
	22	F	70
	23	G	71
	24	H	72
	25	I	73
	26	J	74
	27	K	75
RED	28	L	76
CRSR →	29	M	77
GRN	30	N	78
BLU	31	O	79
SPACE	32	P	80
!	33	Q	81
"	34	R	82
#	35	S	83
$	36	T	84
%	37	U	85
&	38	V	86
'	39	W	87
(	40	X	88
)	41	Y	89
*	42	Z	90
+	43	[	91
,	44	£	92
-	45	]	93
.	46	↑	94
/	47	←	95

Fig. 1 – Tabella dei codici ASCII Commodore.

CARATTERI	CODICE ASCII	CARATTERI	CODICE ASCII
	96	BLK	144
	97	↑ CRSR	145
	98	RVS OFF	146
	99	CLR HOME	147
	100	INST DEL	148
	101		149
	102		150
	103		151
	104		152
	105		153
	106		154
	107		155
	108	PUR	156
	109	← CRSR	157
	110	YEL	158
	111	CYN	159
	112	SPACE	160
	113		161
	114		162
	115		163
	116		164
	117		165
	118		166
	119		167
	120		168
	121		169
	122		170
	123		171
	124		172
	125		173
π	126		174
	127		175
	128		176
	129		177
	130		178
	131		179
	132		180
f1	133		181
f3	134		182
f5	135		183
f7	136		184
f2	137		185
f4	138		186
f6	139		187
f8	140		188
SHIFT RETURN	141		189
SWITCH TO UPPERCASE	142		190
	143		191

Fig. 2 – Tabella dei codici ASCII Commodore.

- I codici da 128 a 160 in parte sono liberi e in parte corrispondono a colori e tasti funzionali.

- Infine, i codici compresi fra 161 e 191 sono relativi ai rimanenti caratteri grafici ROM, attivati premendo contemporaneamente il tasto C= .

Riassumendo quanto sinora precisato, si può asserire che i gruppi di codici 0 ÷ 32 e 128 ÷ 160 non corrispondono a caratteri stampabili, mentre tutti gli altri codici si riferiscono a caratteri stampabili.

Si può ancora fare notare che il codice numerico 14 consente di passare dalla raffigurazione di lettere maiuscole a quella di lettere minuscole, ottenendo cosí lo stesso effetto che si consegue quando si premono contemporaneamente i tasti SHIFT e C= .

Il codice 142 permette di ritornare dalla raffigurazione di lettere minuscole a quella di lettere maiuscole.

Il codice 147 provvede a cancellare lo schermo ed a posizionare il cursore in alto a sinistra sullo schermo stesso, proprio come avviene quando si premono contemporaneamente i tasti SHIFT e CLR | HOME .

Per conoscere il codice ASCII di un determinato carattere o simbolo riportato sulla tastiera, come già precisato nella lezione *Analisi e programmazione 11*, è sufficiente scrivere l'istruzione che segue:

PRINT ASC ("X")

dove X è il carattere di cui si desidera conoscere il codice ASCII.

Per compiere l'operazione inversa, ossia ottenere il carattere dal corrispondente codice ASCII, occorre scrivere l'istruzione:

PRINT CHR$ (N)

dove N è il codice del carattere.

Nella *fig. 3* è riportato il listato di un programma che consente di visualizzare sullo schermo tutti i caratteri stampabili con il computer, i quali, come già accennato prima, sono compresi fra i due gruppi di codici ASCII 33 ÷ 127 e 161 ÷ 191.

Esaminiamo ora le principali fasi operative del programma in questione.

```
10 REM*CARATERI STAMPABILI*
20 REM*   CODICE ASCII     *
30 REM*    COMMODORE       *
40 PRINT CHR$(147)
50 PRINT"CODICE";
60 PRINT"■■■CARATTERE"
70 FOR N=33 TO 127
80 PRINT N,CHR$(N):PRINT
90 FOR T=1 TO 500:NEXT T
100 NEXT N
110 FOR N=161 TO 191
120 PRINT N,CHR$(N):PRINT
130 FOR T=1 TO 500:NEXT T
140 NEXT N
150 END
```

Fig. 3 – Programma che consente di visualizzare tutti i caratteri stampabili del codice ASCII Commodore.

L'istruzione 4Ø provvede a cancellare lo schermo ed a posizionare il cursore in alto a sinistra (codice ASCII 147).

Le istruzioni 5Ø e 6Ø visualizzano sullo schermo le parole CODICE e CARATTERE, quest'ultima preceduta da tre spostamenti del cursore a destra (□ □ □).

L'istruzione 7Ø imposta un'iterazione che fornisce alla variabile N i valori compresi fra 33 e 127, corrispondenti al primo gruppo dei codici ASCII, che vengono inseriti nella funzione CHR$ per ottenere sullo schermo la visualizzazione dei relativi caratteri stampabili (istruzione 8Ø).

Il ciclo di ritardo (istruzione 9Ø) consente di rallentare lo spostamento (scrolling) dei codici ASCII e dei relativi caratteri visualizzati.

In modo analogo, il ciclo **FOR... NEXT...**, presente nelle istruzioni 11Ø e 14Ø, fornisce alla variabile N i valori compresi fra 161 e 191, corrispondenti al secondo gruppo di codici ASCII, che vengono inseriti nella funzione CHR$ per ottenere la stampa dei relativi caratteri (istruzione 12Ø).

Provveda ora a cancellare la memoria del computer battendo il comando **NEW** ed esegua il programma scrivendo il comando **RUN**.

Sulla parte sinistra dello schermo vedrà apparire, con un movimento diretto verso l'alto abbastanza lento, prima il codice ASCII e poi a destra il corrispondente carattere.

Immettendo nel computer la seguente istruzione:

45 **PRINT CHR$** (14)

ed eseguendo nuovamente il programma, le lettere dell'alfabeto (codici 65÷90) verranno visualizzate in forma minuscola. Inoltre, in corrispondenza dei codici 97÷122 saranno raffigurate le lettere maiuscole dell'alfabeto anziché i caratteri grafici ROM.

Conclusa la prova, per ritornare nuovamente nella condizione standard (lettere maiuscole) occorre premere contemporaneamente i tasti **SHIFT** e **C=**.

Il programma riportato nella *fig. 4* permette di vedere sullo schermo i codici ASCII in corrispondenza dei caratteri della tastiera battuti.

Come può osservare, nel programma in questione viene utilizzata la funzione stringa **ASC**("X"), dove con X, come già precisato, viene indicato un tasto qualsiasi della tastiera.

```
10 REM*CODICE ASCII COMMODORE*
15 PRINT"■"
20 PRINT"BATTI UN SIMBOLO"
25 PRINT"DI TASIERA"
30 PRINT
35 INPUT C$
40 PRINT:PRINT"IL CODICE ASCII"
45 PRINT"IN SISTEMA DECIMALE"
50 PRINT"DEL CARATTERE BATTUTO"
55 PRINT"E'";ASC(C$)
57 PRINT:PRINT
60 PRINT"PER PROSEGUIRE BATTI"
63 PRINT"IL TASTO RETURN"
65 PRINT"PER INTERROMPERE"
70 PRINT"BATTI NN"
75 INPUT N$
80 IF N$="NN" THEN END
85 GOTO 15
```

Fig. 4 – Programma per visualizzare i codici dei tasti della tastiera.

Immetta il programma della *fig. 4* nel computer e lo esegua; sullo schermo comparirà il seguente messaggio:

```
BATTI UN SIMBOLO
DI TASTIERA
?
```

A questo punto, se per esempio batte il tasto 1, sullo schermo verrà visualizzato:

```
IL CODICE ASCII
IN SISTEMA DECIMALE
DEL CARATTERE BATTUTO
E': 49
```

Occorre tenere presente che, per immettere nel computer i caratteri separatori virgola (,) e due punti (:), al fine di ottenere il corrispondente codice ASCII, si deve batterli facendoli precedere e seguire dai caratteri virgolette, ossia nel seguente modo:

```
","        ":"
```

Il programma riportato nella *fig. 5* consente invece di vedere sullo schermo i caratteri simbolici della tastiera corrispondenti ai codici numerici inseriti dalla stessa tastiera.

```
10 REM*CARATTERI TASTIERA*
15 REM*    COMMODORE     *
20 PRINT"■■BATTI UN CODICE ASCII"
30 PRINT
35 INPUT C%
40 PRINT:PRINT"IL CARATTERE CORRISPONDENTE"
45 PRINT"AL CODICE";C%;"E':";CHR$(C%)
50 PRINT:PRINT
55 PRINT"PER PROSEGUIRE BATTI"
60 PRINT"IL TASTO RETURN"
65 PRINT"PER INTERROMPERE"
70 PRINT"BATTI NN"
75 INPUT N$
80 IF N$="NN"THEN END
85 GOTO 20
```

Fig. 5 – Programma che consente di visualizzare i simboli o caratteri corrispondenti ai codici ASCII battuti sulla tastiera.

Infatti, come può rilevare, viene utilizzata la funzione **CHR$**, la quale serve appunto a rappresentare all'esterno i codici numerici ASCII contenuti nella memoria del computer.

Se si utilizza il VIC-20, l'istruzione 4Ø dovrà essere cosí modificata:

```
40 PRINT:PRINT "IL CARATTERE CORRISP."
```

Immetta nel computer il programma in questione e lo esegua; sullo schermo comparirà la scritta:

```
BATTI UN CODICE ASCII
?
```

Battendo, ad esempio, il codice 115, sullo schermo apparirà il messaggio:

```
IL CARATTERE CORRISPONDENTE
AL CODICE 115 E' ♥
```

Non tutti i codici ASCII naturalmente possono essere "esternati" sul video con un simbolo o carattere. Ad esempio, battendo il codice ASCII 13, Lei dovrebbe vedere il tasto **RETURN** . Quest'ultimo però, essendo un tasto operativo, non può essere visualizzato ed al suo posto apparirà uno spazio.

2. – Esercitazioni con le stringhe

Come è noto, il computer consente di operare non soltanto con i numeri ma anche con sequenze di caratteri, ossia con le stringhe.

Mediante apposite parole chiavi si può ottenere il contenuto numerico di una stringa, misurarne la lunghezza, suddividerla, estrarre da essa un carattere o una sequenza di caratteri, confrontare e concatenare piú stringhe.

Vediamo ora, con appositi esercizi di programmazione, di verificare l'operatività delle funzioni stringa utilizzate nel linguaggio BASIC.

2.1 – Esercizi con le funzioni VAL e STR$

La funzione **VAL** (A$) consente di convertire la stringa A$ in un numero; la funzione **STR$** opera invece in modo opposto, ossia cambia un numero in una stringa.

Provi ad eseguire, in esecuzione immediata, le seguenti istruzioni:

1) **PRINT VAL** ("SCUOLA")
2) **PRINT VAL** ("-12.33.42.5")
3) **PRINT VAL** ("124, 125")

Relativamente all'istruzione 1) verrà visualizzato il semplice valore Ø (zero), perché la stringa "SCUOLA" inizia con una lettera che non può essere interpretata come numero (si tratta cioè di un'interpretazione "FALSA", che il calcolatore segnala con il valore Ø).

L'istruzione 2) visualizzerà il solo valore numerico -12.33, perché i rimanenti caratteri .42.5 non possono essere interpretati come facenti parte di un numero che già contiene un punto decimale.

L'istruzione 3) farà vedere il numero 124, perché nella stringa "124, 125" è presente il carattere separatore virgola (,) e di conseguenza soltanto le prime tre cifre numeriche possono essere interpretate come numero.

Provi adesso ad eseguire, sempre in esecuzione immediata, le seguenti istruzioni BASIC:

1) **PRINT STR$** (1.23)
2) **PRINT STR$** (A)
3) **PRINT STR$** (-Ø.5)
4) **PRINT STR$** ("1.23")

Sullo schermo nel caso 1) apparirà il numero-stringa 1.23 (cioè cosí come è scritto nell'istruzione **STR$**); nel caso 2) apparirà il semplice numero-stringa Ø, perché la lettera A interpretata come numero vale Ø; nel caso 3) comparirà il numero-stringa -Ø.5.

Nel caso 4) verrà segnalato errore (TYPE MISMATCH ERROR), perché l'argomento "1.23", all'interno delle parentesi, deve essere una costante numerica e non una costante stringa, cioè dentro le parentesi della funzione stringa **STR$** ci possono essere solamente costanti o variabili numeriche.

2.2 – Esercizi con le funzioni **LEN** (X$), **LEFT$** (X$,X), **MID$** (X$,S,X), **RIGHT$** (X$,X)

Il semplice programma riportato nella *fig. 6* consente di calcolare il numero di caratteri costituenti qualsiasi stringa X$ immessa nel computer.

Ciò si ottiene utilizzando la funzione **LEN** (X$), (istruzione 45), che fornisce appunto la lunghezza in caratteri di una generica stringa.

Questa funzione risulta particolarmente utile, ad esempio, quando si desidera conoscere se una data stringa può essere contenuta in una delimitata zona dello schermo.

```
10 REM*LUNGHEZZA DI UNA *
20 REM*     STRINGA      *
30 PRINT"J"
35 PRINT"IMMETTERE LA STRINGA"
40 INPUT S$
45 X=LEN(S$)
47 PRINT
50 PRINT "LA LUNGHEZZA DELLA"
55 PRINT"STRINGA E' DI";X;"CARATTERI"
57 PRINT
60 PRINT"PER PROSEGUIRE BATTI"
65 PRINT"IL TASTO RETURN"
70 PRINT"PER FINIRE BATTI NN"
75 INPUT N$
80 IF N$="NN" THEN END
85 GOTO 30
```

Fig. 6 – Programma che calcola la lunghezza (numero dei caratteri) di una stringa.

Immetta il programma della *fig. 6* nel computer e lo esegua; sullo schermo comparirà il messaggio:

```
IMMETTERE LA STRINGA
?
```

A questo punto può battere la stringa di cui desidera conoscere il numero di caratteri, per esempio la seguente:

```
12 OTTOBRE 1492
```

Il computer risponderà:

```
LA LUNGHEZZA DELLA
STRINGA E' DI 15 CARATTERI
```

Infatti, vengono anche considerati caratteri i due spazi liberi dopo il numero 12 e dopo la parola OTTOBRE.

Il programma riportato nella *fig. 7*, il quale utilizza la funzione **LEN** insieme alla funzione **LEFT$**, permette invece di visualizzare sullo schermo l'anagramma triangolare della parola "TAVOLA".

La funzione **LEFT$**, come già specificato nella lezione *Analisi e programmazione 11*, estrae dalla stringa il numero di caratteri specificati in argomento, partendo dall'estremità sinistra della stringa stessa.

```
10 REM*ANAGRAMMA DI*
12 REM*UNA STRINGA *
15 PRINT"□"
20 N=LEN("TAVOLA")
30 FOR I=1 TO N
40 PRINT LEFT$("TAVOLA",I)
50 NEXT
60 END
```

Fig. 7 – Programma che visualizza l'anagramma triangolare di una stringa.

L'istruzione 20 del programma della *fig. 7* calcola e memorizza in N il numero 6, corrispondente al numero di caratteri della parola in esame (tavola).

Questo numero serve al successivo ciclo **FOR... NEXT...** per individuare e visualizzare in successione il primo carattere, i primi due caratteri, i primi tre caratteri, ecc. posti a sinistra della parola.

Tale ciclo finisce quando sono stati visualizzati tutti i sei caratteri della parola "TAVOLA".

La funzione stringa che consente di individuare i primi caratteri di una parola è la **LEFT$**.

Scrivendo quindi

LEFT$ ("TAVOLA", I)

il computer individua e rende disponibili per la stampa i primi I caratteri posti a sinistra della parola: se I=1, viene individuata la lettera T; se I=2, vengono individuate le lettere TA; se I=3, vengono individuate le lettere TAV; ecc.

Dopo aver riportato il computer nelle condizioni iniziali mediante il comando **NEW** ed i tasti RUN | STOP e RESTORE, batta il listato del programma della *fig. 7* e lo esegua.

Sullo schermo comparirà l'anagramma che segue:

T
TA
TAV
TAVO
TAVOL
TAVOLA

Provi ora a sostituire nel programma l'istruzione 40 con la seguente:

40 PRINT RIGHT$ ("TAVOLA",I)

e riesegua il programma stesso. Sullo schermo apparirà quest'altro anagramma della parola TAVOLA:

A
LA
OLA
VOLA
AVOLA
TAVOLA

Ciò è dovuto alla funzione **RIGHT$**, con la quale vengono individuati gli ultimi I caratteri posti piú a destra della stringa "TAVOLA".

Un altro semplice esempio di come si può scomporre (segmentare) una stringa è evidenziato nel programmino che segue.

```
10 REM*SCOMPORRE UNA STRINGA*
20 S$="ABCDEFGHILMNOPQRSTUVZ"
30 PRINT LEFT$(S$,7)
40 PRINT MID$(S$,8,7)
50 PRINT RIGHT$(S$,7)
60 END
```

Questo semplicissimo programma provvede a suddividere la stringa S$, costituita dalle 21 lettere dell'alfabeto, in tre parti, denominate **sottostringhe**, ognuna delle quali è costituita da sette lettere.

In particolare, l'istruzione 3Ø preleva le prime sette lettere, iniziando da sinistra, della stringa S$; l'istruzione 4Ø, con la funzione **MID$**, provvede, partendo dall'ottava lettera, ad estrarre anch'essa sette lettere della stringa S$; infine l'istruzione 5Ø, partendo dall'estremità destra della stringa S$, estrae le ultime sette lettere dell'alfabeto.

Scriva ed esegua il programma in questione; sullo schermo compariranno le seguenti tre sottostringhe:

```
ABCDEFG
HILMNOP
QRSTUVZ
```

Vediamo ora come si possono memorizzare i colori riportati sui tasti del computer in una stringa di nome C$ e come possono essere scelti e prelevati in modo del tutto casuale per andare a "colorare" i caratteri di un'altra stringa che apparirà sullo schermo.

Possiamo scegliere come stringa da colorare l'espressione:

"SCUOLA RADIO ELETTRA"

costituita da 2Ø caratteri, compresi gli spazi. Si tratta di colorare questi 2Ø caratteri in modo del tutto casuale.

A questo scopo, prima di visualizzare un carattere dell'intera stringa "SCUOLA RADIO ELETTRA" sullo schermo, si può calcolare con la funzione **RND** (1) un numero a caso compreso tra 1 e 7, che serve a scegliere uno dei sette colori memorizzati nella stringa C$ (il colore bianco non viene usato perché non visibile sullo schermo anch'esso bianco).

Nella *fig. 8* è rappresentato il flow-chart del programma in questione, mentre nella *fig. 9* e nella *fig. 10* sono riportati i corrispondenti listati, relativi rispettivamente al C64 ed al VIC-20.

Come si vede dalla codifica della *fig. 9* e della *fig. 10*, le istruzioni 15÷2Ø schiariscono lo schermo ed impostano la successiva stampa alla riga otto: ciò viene ottenuto battendo sette volte consecutivamente il tasto ⇑ CRSR ⇓ " Q Q Q Q Q Q Q ".

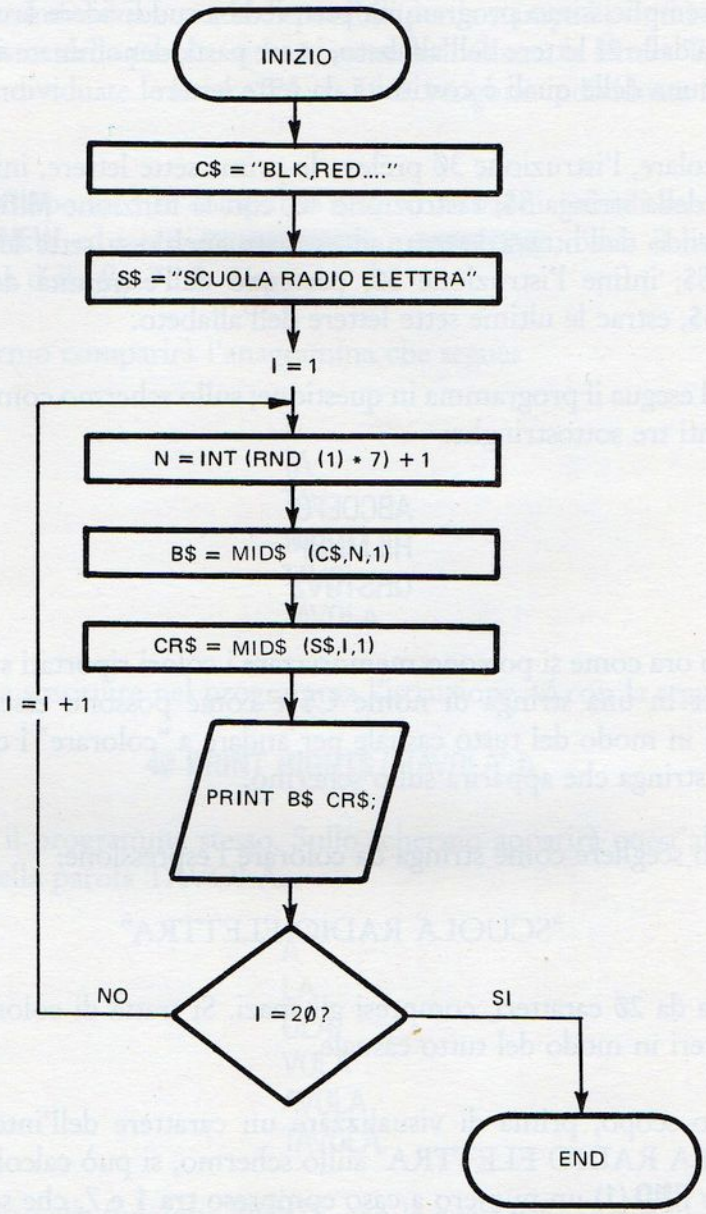


Fig. 8 – Diagramma di flusso del programma che visualizza la scritta SCUOLA RADIO ELETTRA con i relativi caratteri colorati in modo casuale.

Si può inoltre notare che nel listato della *fig. 9* (relativo al C64) l'istruzione 17 provvede a colorare di bianco lo schermo per far risaltare meglio i colori della stringa di caratteri.

L'istruzione 25 assegna alla variabile stringa di nome C$ i sette colori con cui colorare i caratteri. Questi colori vengono ottenuti mantenendo premuto il tasto **CTRL** e contemporaneamente battendo in successione i tasti **BLK**, **RED**, **CYN**, **PUR**, **GRN**, **BLU** e **YEL**.

```
10 REM*SCRITTA A COLORI*
15 PRINT"■"
17 POKE 53281,1
20 PRINT"■■■■■■■■"
25 C$="■■■■■■■"
30 S$="SCUOLA RADIO ELETTRA"
35 FOR I=1 TO 20
40 N=INT(RND(1)*7)+1
45 B$=MID$(C$,N,1)
50 CR$=MID$(S$,I,1)
55 PRINT B$;CR$;
60 NEXT I
65 PRINT:PRINT:PRINT:PRINT:PRINT
70 END
```

Fig. 9 – Programma che visualizza sullo schermo del C64 la scritta SCUOLA RADIO ELETTRA con i relativi caratteri colorati in modo del tutto casuale.

```
10 REM*SCRITTA A*
12 REM* COLORI  *
15 PRINT"■"
20 PRINT"■■■■■■■■■"
25 C$="■■■■■■■"
30 S$="SCUOLA RADIO ELETTRA"
35 FOR I=1 TO 20
40 N=INT(RND(1)*7)+1
45 B$=MID$(C$,N,1)
50 CR$=MID$(S$,I,1)
55 PRINT B$;CR$;
60 NEXT I
65 PRINT:PRINT:PRINT:PRINT:PRINT
70 END
```

Fig. 10 – Programma che visualizza sullo schermo del VIC-20 la scritta SCUOLA RADIO ELETTRA con i relativi caratteri colorati in modo del tutto casuale.

L'istruzione 3Ø assegna la stringa "SCUOLA RADIO ELETTRA" alla variabile di nome S$.

Nel ciclo di programma 35÷6Ø si calcola ogni volta un numero a caso compreso fra 1 e 7, numero che viene utilizzato per scegliere uno dei sette colori depositati nella stringa C$.

Successivamente, con il colore appena scelto, viene colorata e visualizzata sullo schermo una lettera della stringa "SCUOLA RADIO ELETTRA"; tale lettera viene scelta in ordine crescente a partire dall'inizio dell'espres-

sione, cioè nell'ordine: la prima, la seconda, la terza, ..., la ventesima ed ogni lettera viene stampata di seguito alla precedente.

L'istruzione 55 di stampa termina infatti con il carattere (;), che, come è noto, consente di concatenare piú stringhe in una stessa linea di stampa.

Il programma termina quando tutta l'espressione (scritta) risulta completa sullo schermo.

L'istruzione 65, costituita da piú **PRINT**, serve a distanziare dalla stampa a colori il successivo messaggio **READY** (pronto).

Immetta ora nel computer il programma descritto e lo esegua; sullo schermo comparirà la scritta a colori:

SCUOLA RADIO ELETTRA

Rieseguendo lo stesso programma (con il comando **RUN**), tale scritta apparirà colorata in un altro modo, perché la funzione RND sceglierà sempre in modo diverso i numeri (cioè i colori) da 1 a 7.

Come ultimo esercizio sull'uso delle stringhe eseguiremo un programma, formulato sotto forma di indovinello, il quale permette di evidenziare la possibilità offerta dal linguaggio BASIC di effettuare il confronto fra stringhe per accertare se sono eguali.

Il flow-chart del programma in questione è rappresentato nella *fig. 11* e nella *fig. 12* è riportata la relativa codifica.

Le linee di programma 1Ø e 15 visualizzano sullo schermo il nome dei sei colori da indovinare.

La linea di programma 2Ø provvede ad assegnare il valore 1 alla variabile A (contatore).

La linea 25 genera un numero a caso compreso fra 1 e 6 (funzione **RND**), il quale viene utilizzato per selezionare uno dei sei colori da indovinare per effetto dell'istruzione **ON... GOTO...**, la quale permette di effettuare salti incondizionati alle istruzioni 3Ø, 4Ø, 5Ø, 6Ø, 7Ø e 8Ø in cui sono depositati i nomi dei colori (stringhe C$).

L'istruzione 12Ø avverte l'operatore che può immettere nel computer il nome del colore da lui stesso prescelto (stringa A$) fra i sei proposti.

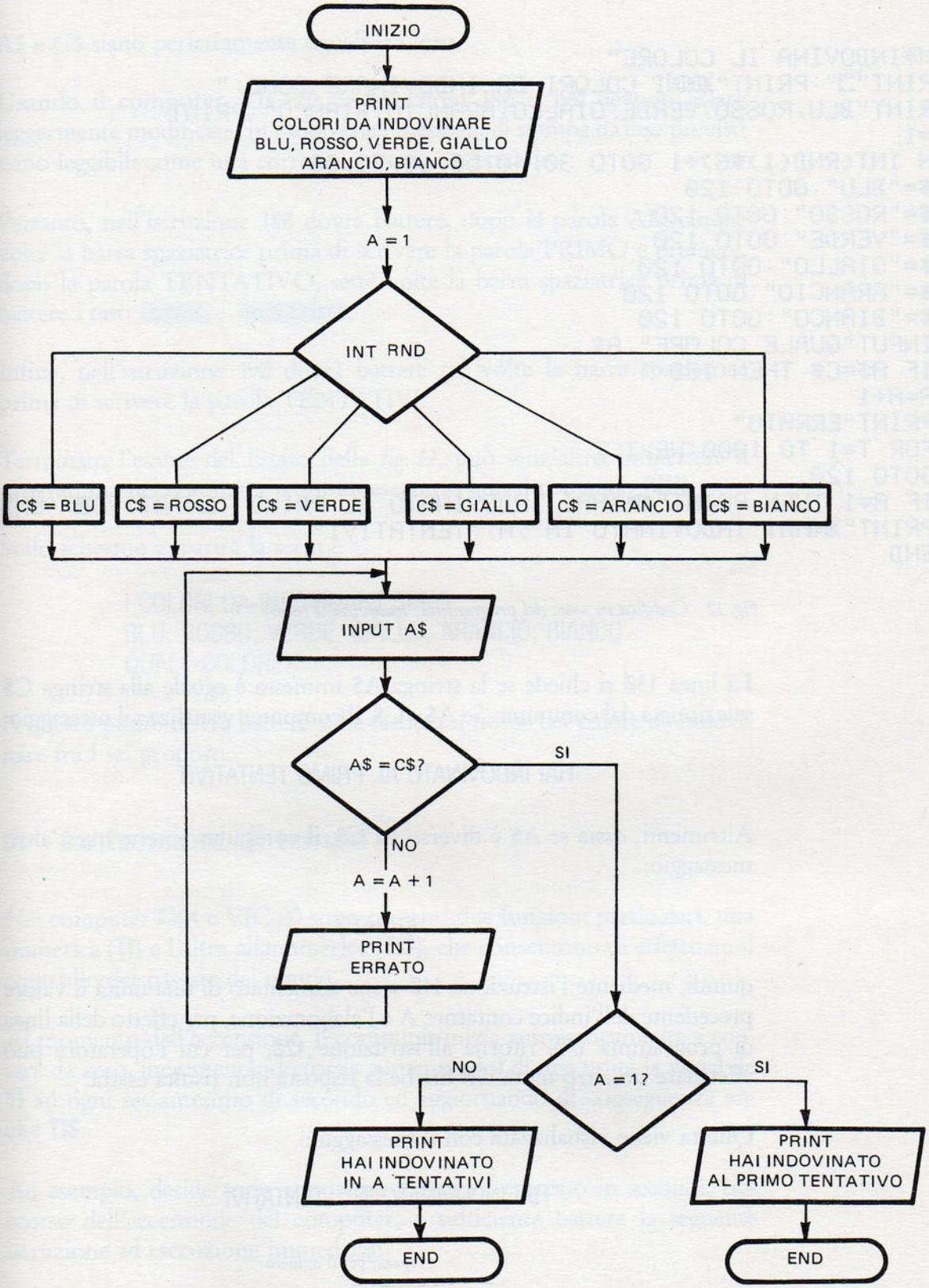


Fig. 11 – Flow-chart del programma "Indovina il colore".

```
5 REM*INDOVINA IL COLORE"
10 PRINT"□":PRINT"■■■I COLORI DA INDOVINARE SONO:"
15 PRINT"BLU,ROSSO,VERDE,GIALLO,ARANCIO,BIANCO":PRINT
20 A=1
25 ON INT(RND(1)*6)+1 GOTO 30,40,50,60,70,80
30 C$="BLU":GOTO 120
40 C$="ROSSO":GOTO 120
50 C$="VERDE":GOTO 120
60 C$="GIALLO":GOTO 120
70 C$="ARANCIO":GOTO 120
80 C$="BIANCO":GOTO 120
120 INPUT"QUALE COLORE";A$
130 IF A$=C$ THEN 180
140 A=A+1
150 PRINT"ERRATO"
160 FOR T=1 TO 1000:NEXT T
170 GOTO 120
180 IF A=1 THEN PRINT"■■■HAI INDOVINATO AL PRIMO TENTATIVO■":END
190 PRINT"■■HAI INDOVINATO IN";A;"TENTATIVI"
200 END
```

Fig. 12 – Codifica in BASIC del programma "Indovina il colore".

La linea 13Ø si chiede se la stringa A$ immessa è eguale alla stringa C$ selezionata dal computer. Se A$=C$, il computer visualizza il messaggio:

HAI INDOVINATO AL PRIMO TENTATIVO

Altrimenti, ossia se A$ è diverso da C$, il computer emette quest'altro messaggio:

ERRATO

quindi, mediante l'istruzione 14Ø viene aumentato di una unità il valore precedente dell'indice contatore A e l'elaborazione, per effetto della linea di programma 17Ø, ritorna all'istruzione 12Ø, per cui l'operatore può effettuare un altro tentativo finché la risposta non risulta esatta.

Questa viene visualizzata con il messaggio:

HAI INDOVINATO IN TENTATIVI

↑

spazio per il numero

Per indovinare il colore è necessario che tutte le lettere delle due stringhe

A$ e C$ siano perfettamente uguali tra loro.

Usando il computer VIC-20, le istruzioni 18Ø e 19Ø devono essere leggermente modificate, in modo che i messaggi di stampa da esse previsti siano leggibili come una corretta espressione letterale.

Pertanto, nell'istruzione 18Ø dovrà battere, dopo la parola AL, cinque volte la barra spaziatrice prima di scrivere la parola PRIMO e premere, dopo la parola TENTATIVO, sette volte la barra spaziatrice prima di battere i tasti CTRL e RVS | OFF.

Infine, nell'istruzione 19Ø dovrà battere tre volte la barra spaziatrice prima di scrivere la parola TENTATIVI.

Terminato l'esame del listato della *fig. 12*, può senz'altro immettere il programma nel computer e quindi eseguirlo con il comando **RUN**.

Sullo schermo apparirà la scritta:

```
I COLORI DA INDOVINARE SONO
BLU, ROSSO, VERDE, GIALLO, ARANCIO, BIANCO
QUALE COLORE?
```

A questo punto dovrà battere sulla tastiera il nome del colore da indovinare fra i sei proposti.

3. - Le funzioni di tempo

Nei computer C64 e VIC-20 sono presenti due funzioni particolari, una numerica (**TI**) e l'altra alfanumerica (**TI$**), che consentono di effettuare il controllo del passare del tempo.

Al momento dell'accensione, il computer inizia automaticamente a contare da zero, incrementando (ossia aumentando) di una unità la variabile **TI** ad ogni sessantesimo di secondo ed aggiornando di conseguenza anche **TI$**.

Ad esempio, desiderando conoscere il tempo, espresso in secondi, trascorso dell'accensione del computer, è sufficiente battere la seguente istruzione ad esecuzione immediata:

```
PRINT TI/60
```

Per verificare l'operatività della funzione **TI** possiamo ora eseguire un breve programma, che consente di determinare il tempo impiegato dal computer per realizzare un ciclo di ritardo (pausa temporale) costituito da una istruzione **FOR... NEXT...**, il cui valore finale di conteggio è pari a 3ØØØ.

Il listato del programma in questione è riportato nella *fig. 13*.

```
10 REM*PROVA FUNZIONE*
15 REM*  DI TEMPO TI *
20 PRINT"□"
25 S=TI
30 FOR T=1 TO 3000:NEXT T
35 X=TI-S
40 PRINT"TEMPO IMPIEGATO:"
45 PRINT X/60"SEC."
50 END
```

Fig. 13 – Programma che fornisce il tempo di esecuzione di una istruzione **FOR... NEXT...** *di 3ØØØ cicli.*

L'istruzione 25 provvede ad assegnare alla variabile di nome S la funzione **TI**.

Nella riga di programma 3Ø è presente l'iterazione **FOR... NEXT...** di cui si desidera conoscere il tempo di esecuzione.

L'istruzione 35 assegna alla variabile X il valore TI-S, che diviso per 6Ø (istruzione 45) fornisce il tempo, espresso in secondi, impiegato dalla **FOR... NEXT...** costituita da 3ØØØ cicli.

Immetta ora il programma nel computer e lo esegua. Sullo schermo comparirà il numero indicante i secondi impiegati dal computer per eseguire l'istruzione della riga 3Ø.

Potrà eventualmente aumentare o diminuire il valore finale di conteggio dell'istruzione **FOR... NEXT...** (riga 3Ø) per verificare le corrispondenti variazioni dei tempi d'esecuzione.

La funzione alfanumerica **TI$** è composta invece da sei caratteri: i primi due contengono il numero delle ore, il terzo ed il quarto il numero dei minuti, il quinto ed il sesto il numero dei secondi.

Ai caratteri di questa funzione può essere attribuito qualsiasi valore (dal

momento che essi sono caratteri numerici).

La funzione **TI$** può quindi funzionare da orologio, come Lei stesso può verificare eseguendo il programma riportato nella *fig. 14*.

Nell'istruzione 2Ø di questo programma è necessario scrivere, al posto di "Ø Ø Ø Ø Ø Ø", i valori relativi all'ora, ai minuti e ai secondi da Lei prescelti, i quali verranno automaticamente aggiornati e riprodotti sullo schermo mediante l'istruzione 4Ø.

```
10 REM*PROVA FUNZIONE*
12 REM* DI TEMPO TI$ *
15 PRINT"■"
20 TI$="000000"
25 PRINT"■■ORA ESATTA"
30 PRINT"■"
40 PRINT"■■■■■■■■■■■"TI$"■";:GOTO 30
45 END
```

Fig. 14 – Programma che fornisce l'ora.

Per esempio, per predisporre l'orologio alle ore 12:20 minuti e 15 secondi (antimeridiane), si dovrà scrivere l'istruzione 2Ø nel seguente modo:

```
20 TI$="122015"
     ore  minuti  secondi
```

Eseguendo il programma, verranno visualizzati sullo schermo l'ora, i minuti e i secondi a partire dai valori da Lei immessi nella stringa **TI$**.

Se si lascia acceso il computer per più di ventiquattro ore, esso viene automaticamente azzerato.

Passiamo ora ad analizzare il modo in cui si ottiene l'animazione degli sprite sul C64.

Quindi completeremo l'esame delle diverse possibilità offerte dai computer C64 e VIC-20 nel campo musicale.

A seconda del computer in Suo possesso (C64 oppure VIC-20), **dovrà perciò fare riferimento al relativo capitolo che troverà di seguito.**

4. - Animazione dello sprite con il C64

Per ottenere l'effetto di animazione dello sprite, occorre elaborare due blocchi di dati ad esso relativi, i quali differiscano fra loro soltanto per alcuni valori, al fine di creare appunto sullo schermo l'illusione di una figura in movimento.

In particolare, se per memorizzare il primo blocco di dati relativo allo sprite da visualizzare viene utilizzata l'area di memoria 832 (13×64=832), come già fatto nei precedenti esercizi di programmazione, il secondo blocco di dati, corrispondente sempre allo stesso sprite, verrà memorizzato nell'area di memoria successiva, ad esempio nell'area 896 (14×64=896).

Ovviamente, nel programma dovrà essere previsto un secondo ciclo di lettura e memorizzazione dei dati per immettere nella memoria del computer il secondo blocco dei dati dello sprite.

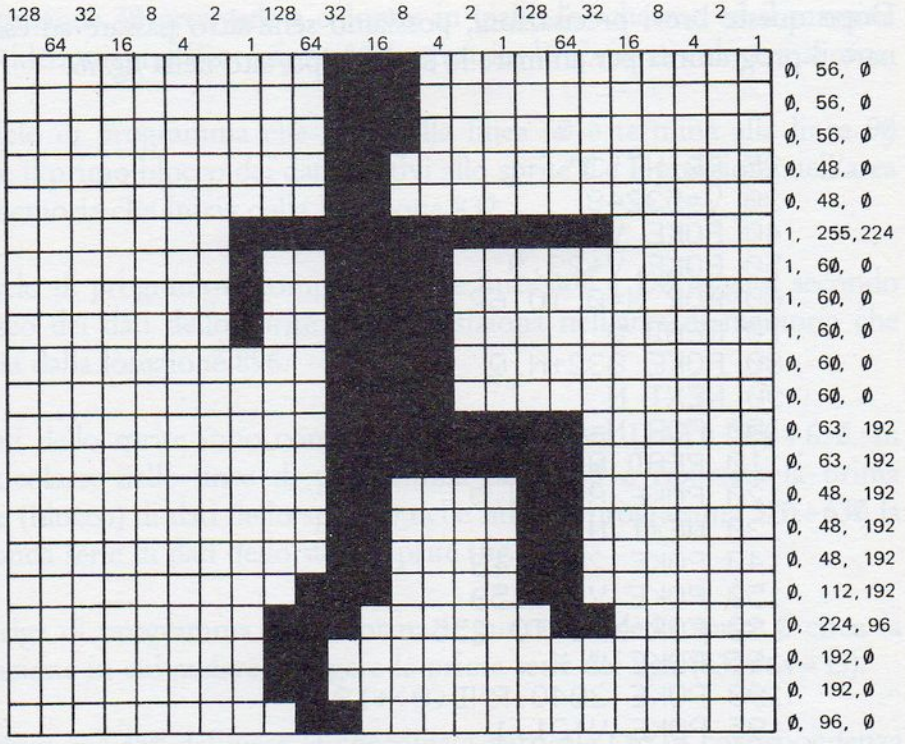
L'esercizio di programmazione che ci accingiamo ad eseguire, il quale visualizzerà uno sprite raffigurante un uomo in corsa che si sposta dal lato sinistro al lato destro dello schermo, permetterà di chiarire facilmente quanto sinora precisato.

Gli schemi dei dati relativi allo sprite raffigurante un uomo che corre sono riportati nella *fig. 15* e sono stati ricavati con il solito sistema già precedentemente descritto.

Per ottenere il completo spostamento dello sprite dal lato sinistro al lato destro dello schermo verrà utilizzata l'apposita locazione di memoria del VIC-II, contraddistinta dal numero 53264 (V+16), la quale controlla appunto lo spostamento dell'immagine (sprite) in un punto orizzontale dello schermo compreso fra i valori 256 e 32Ø.

Come certamente ricorderà, senza l'uso di tale locazione di memoria l'immagine può spostarsi soltanto in corrispondenza della posizione 255, poiché il registro dello spostamento X (orizzontale dello sprite) può contenere un valore numerico massimo pari a 255.

I valori da inserire nella locazione V+16 (X>255) devono essere compresi fra Ø e 63, dal momento che sono soltanto 64 le posizioni che possono essere contenute fra i valori 256 e 32Ø.



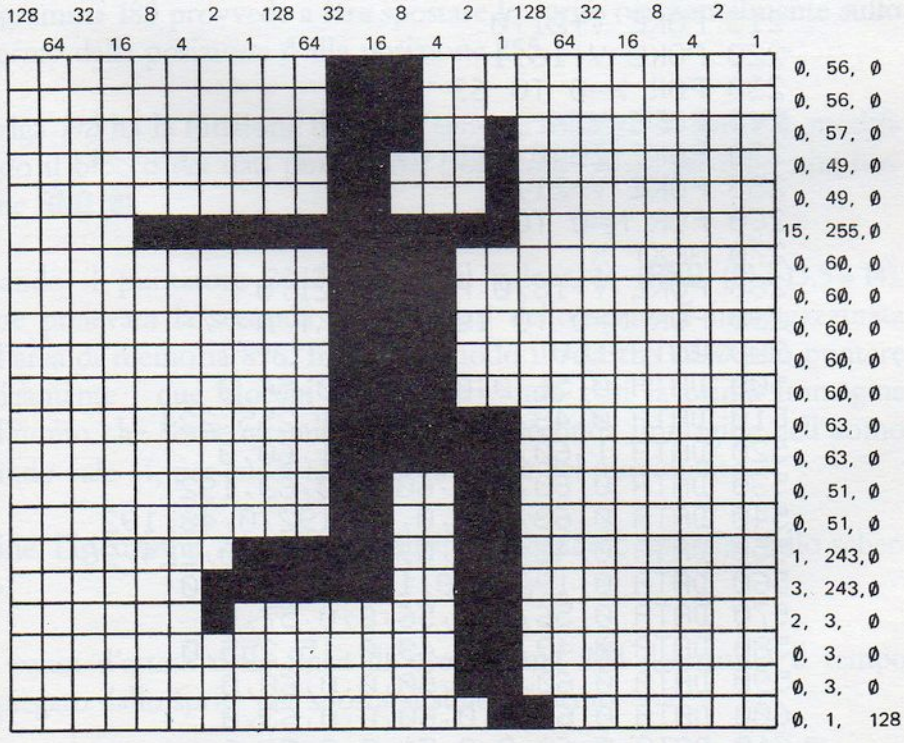


Fig. 15 – Schemi dei dati di uno sprite raffigurante un uomo che corre.

Dopo queste brevi precisazioni, possiamo senz'altro passare ad esaminare il programma per animare lo sprite, riportato nella *fig. 16*.

```
25 PRINT"□"
30 V=53248
40 POKE V+32,5:POKE V+33,5
50 POKE V+39,9
60 FOR N=0 TO 62
70 READ Q
80 POKE 832+N,Q
90 NEXT N
100 FOR N=0 TO 62
110 READ Q
120 POKE 896+N,Q
130 NEXT N
140 POKE 2040,13
150 POKE V+1,150
180 FOR X=0 TO 255
185 POKE V,X
190 POKE 2040,RND(0)+13.5
195 POKE V+21,1
200 FOR N=0 TO 25:NEXT N
210 NEXT X
215 POKE V+21,0
220 POKE V+16,1
230 FOR X=0 TO 63
240 POKE V,X
250 POKE 2040,RND(0)+13.5
255 POKE V+21,1
260 FOR N=0 TO 25 :NEXT N
270 NEXT X
280 POKE V+16,0:POKE V+21,0
290 FOR N=0 TO 1000 :NEXT N
300 GOTO 150
500 DATA 0,56,0,0,56,0,0,56,0
510 DATA 0,48,0,0,48,0,1,255,224
520 DATA 1,60,0,1,60,0,1,60,0
530 DATA 0,60,0,0,60,0,0,63,192
540 DATA 0,63,192,0,48,192,0,48,192
550 DATA 0,48,192,0,112,192,0,224,96
560 DATA 0,192,0,0,192,0,0,96,0
570 DATA 0,56,0,0,56,0,0,57,0
580 DATA 0,49,0,0,49,0,15,255,0
590 DATA 0,60,0,0,60,0,0,60,0
600 DATA 0,60,0,0,60,0,0,63,0
610 DATA 0,63,0,0,51,0,0,51,0
620 DATA 1,243,0,3,243,0,2,3,0
630 DATA 0,3,0,0,3,0,0,1,128
```

Fig. 16 – Listato del programma BASIC *per animare uno sprite.*

L'istruzione 4Ø provvede a colorare in giallo lo schermo ed il margine del video, mentre l'istruzione 5Ø fa apparire lo sprite di colore marrone.

Il ciclo di programma che inizia alla linea 6Ø e termina alla linea 9Ø legge il primo blocco dei dati relativi allo sprite Ø e li posiziona nell'area di memoria che inizia dalla locazione 832.

Il ciclo di programma compreso fre le linee 1ØØ e 13Ø legge il secondo blocco dei dati dello sprite Ø e li posiziona nell'area di memoria che inizia dalla locazione 896.

I dati dello sprite sono contenuti nelle istruzioni DATA 5ØØ÷63Ø. In particolare, nelle linee di programma 5ØØ÷56Ø è riportata la prima serie (blocco) di dati dello sprite e nelle linee di programma 57Ø÷63Ø la seconda serie di dati dello stesso sprite (*fig. 16*).

La riga di programma 14Ø informa il puntatore dello sprite Ø circa la locazione in cui andare a leggere la prima serie dei dati (832/64=13).

L'istruzione 15Ø definisce la coordinata verticale (Y) in corrispondenza della quale verrà visualizzato sullo schermo lo sprite, mentre la linea di programma 18Ø provvede a fare spostare lo sprite orizzontalmente sullo schermo dalla posizione Ø alla posizione 255.

La riga 19Ø ha la funzione di creare sempre soltanto lo sprite Ø, modificando il blocco dei dati puntati dal puntatore (2Ø4Ø) secondo una funzione **RND** (Ø).

Quando il puntatore 2Ø4Ø assume il valore 14 (**RND** (Ø)+13.5=14), viene prelevata la seconda serie di dati dello sprite Ø immagazzinata nell'area di memoria 896. In questo modo il registro 2Ø4Ø può puntare ciclicamente i due blocchi di dati, attivando cosí la prima immagine dell'uomo che corre quando vale 13 e la seconda immagine dell'uomo quando vale 14, per cui si ottiene l'illusione del movimento.

Infine, l'istruzione 195 abilita lo sprite Ø facendolo apparire sullo schermo.

La pausa d'attesa della linea di programma 2ØØ determina il tempo impiegato dallo sprite per spostarsi sullo schermo.

Le linee di programma comprese fra 2ØØ e 27Ø provvedono a fare proseguire lo spostamento orizzontale dello sprite Ø dalla posizione 256

alla posizione 32Ø.

In particolare, la linea 22Ø attiva il registro V+16 (che è appunto preposto a tale funzione), mentre l'istruzione 23Ø stabilisce il numero degli spostamenti compresi fra 256 e 32Ø.

L'istruzione 28Ø provvede ad escludere il registro V+16, in modo che la figura possa nuovamente spostarsi dal margine sinistro dello schermo ed a disattivare lo sprite Ø (V+21,Ø).

Occorre rilevare che, prima di attivare il registro V+21, si devono preparare il registro V+16 ed il registro V con i relativi valori: se questo non viene fatto, si avrà un irregolare funzionamento del programma.

Il ciclo di ritardo della linea 29Ø stabilisce il tempo impiegato dallo sprite per ricomparire sul lato sinistro dello schermo.

Terminato l'esame del listato della *fig. 16*, può passare ad immetterlo nel computer.

Eseguendo il programma con il comando **RUN**, vedrà comparire sullo schermo l'immagine stilizzata di un uomo che corre, spostandosi dal lato sinistro al lato destro.

Volendo ingrandire lo sprite, si devono, come già visto nella precendente lezione, attivare le apposite locazioni V+23 (espansione in verticale) e V+29 (espansione in orizzontale), ossia si deve introdurre nel programma l'istruzione che segue:

145 **POKE** V+23,1: **POKE** V+29,1

Lo sprite che appare sullo schermo si può far muovere in modo che copra con la sua figura eventuali caratteri (immagini) presenti sullo schermo stesso, oppure si può far muovere in modo che risulti nascosto dai caratteri, creando cosí l'illusione che le due immagini (sprite e carattere) si trovino su piani separati (effetto tridimensionale di profondità).

Per ottenere questi due diversi effetti si utilizza la locazione di memoria V+27 (priorità, ossia livello d'importanza, dello sprite sullo sfondo).

Per verificare l'operatività di tale locazione di memoria, immetta nel programma le due istruzioni che seguono:

```
35 POKE V+27,Ø
37 POKE 1564,166: POKE 55796,Ø: POKE 1565,166: POKE 55797,Ø
```

L'istruzione 35 inserisce il valore Ø nel registro V+27, per cui lo sprite copre con la sua figura i due caratteri aventi il codice 166 (▩), inseriti rispettivamente nelle locazioni di memoria dello schermo 1564 e 1565.

Rieseguendo il programma con il comando **RUN**, vedrà appunto chiaramente lo sprite passare **davanti** ai due caratteri coprendoli.

Immettendo invece il valore 1 nella locazione di memoria V+27, ossia scrivendo:

```
35 POKE V+27,1
```

vedrà lo sprite passare **dietro** i due caratteri.

Il programma riportato nella *fig. 16* può essere utilizzato ovviamente per animare qualsiasi altro sprite da Lei creato, sostituendo i valori delle istruzioni **DATA** 5ØØ÷63Ø con quelli relativi alla nuova immagine in movimento.

5. - Suoni ed effetti sonori con il C64

Vediamo ora un programma che consente di riprodurre con il computer qualsiasi brano musicale.

Al di là del puro e semplice esercizio, le applicazioni di un tale programma sono svariate: si possono rendere piú brillanti e piacevoli i giochi realizzati sul sistema, sottolineando le varie fasi con motivetti diversi; si può realizzare una sveglia digitale che ci svegli al mattino al suono di una musichetta anziché con il solito trillo antipatico; si possono generare suoni d'allarme che, viceversa, devono essere sgradevoli ed urtanti per attrarre l'attenzione e cosí via.

Per programmare un brano musicale con il computer C64, occorre in primo luogo codificarlo, ossia passare dai simboli dello spartito musicale (*fig. 8* della precedente lezione) ai valori numerici corrispondenti (*fig. 23* della *Pratica 10*).

Per quanto riguarda la durata delle note, si adotterà la stessa codifica usata dai musicisti e riassunta nella tabella della *fig. 8* della precedente lezione, prendendo però come intervallo elementare un valore numerico compreso generalmente fra 15 e 3Ø, a seconda del particolare tempo musicale del brano o del proprio gusto.

Per esempio, se si adotta il valore numerico 2Ø per la semibiscroma, la biscroma avrà il valore 2*2Ø=4Ø, la semicroma il valore 4*2Ø=8Ø e cosí via di seguito, per cui si avrà per la nota breve il valore numerico 128*2Ø=256Ø.

Cerchiamo ora di mettere in pratica quanto precisato, codificando lo spartito musicale (*fig. 17*), relativo al famoso brano di Beethoven intitolato "Per Elisa".

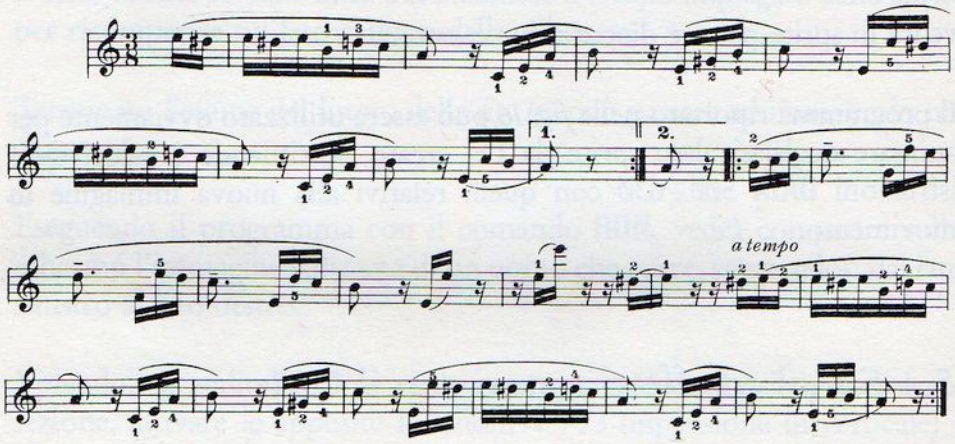


Fig. 17 – Spartito musicale del brano "Per Elisa" di Beethoven.

Il modo piú razionale consiste nello scrivere, uno di seguito all'altro in colonna, i codici (Hi-Byte e Lo-Byte) corrispondenti alle note del brano e nel riportare accanto al codice di ogni nota la relativa durata prescelta.

In questo caso, per quanto concerne la durata delle note, è stato considerato il valore 15,5 per la semibiscroma (intervallo elementare), quindi si hanno le seguenti altre durate, i cui valori sono stati arrotondati:

semibiscroma con punto=22
biscroma=31
biscroma con punto=48
semicroma=62
semicroma con punto=9Ø
croma=125
croma con punto=185
semiminima=25Ø
semiminima con punto=375
minima=5ØØ
minima con punto=75Ø
semibreve=1ØØØ

Naturalmente, il pezzo musicale può essere programmato con un'altra durata delle note, aumentando oppure diminuendo l'intervallo elementare adottato.

La codifica delle prime note del motivo relativo al brano "Per Elisa" è riportata qui di seguito.

CODICE NOTA		DURATA	
Hi-Byte	Lo-Byte		
21	237	62	(semicroma)
2Ø	178	62	
21	237	62	
2Ø	178	62	
21	237	62	
16	109	62	
19	137	62	
17	103	62	
14	162	125	(croma)
Ø	Ø	25Ø	(pausa)
8	18Ø	62	
1Ø	247	62	
14	162	62	
16	1Ø9	125	
⋮	⋮	⋮	

I valori numerici codificati vengono poi immessi nella memoria del computer mediante istruzioni archivio **DATA**.

Come si può rilevare, ciascuna nota è contraddistinta da tre valori numerici.

E' opportuno scrivere soltanto quattro note musicali con relativa durata in ciascuna istruzione **DATA**, al fine di permettere un facile e rapido controllo delle stesse.

L'ultima istruzione **DATA** del programma deve contenere i valori –1, –1, –1, che determinano artificialmente appunto la "fine" del motivo musicale.

Codificato il brano prescelto attenendosi alle indicazioni fornite, passiamo ad esaminare il programma che permette di introdurlo nel computer e di eseguirlo quante volte si desidera.

```
10 REM*PROGRAMMA PER SUONARE*
15 PRINT"■"
20 PRINT TAB(13)"■MUSICA CON C64■"
30 SI=54272
35 VO=54296
40 POKE SI+5,190:POKE SI+6,216
45 POKE VO,15
50 READ FH,FL,D
55 IF FH=-1 THEN END
60 POKE SI,FL:POKE SI+1,FH
65 POKE SI+4,17
70 FOR T=1 TO D:NEXT
80 POKE SI+4,16
85 FOR T=1 TO 50:NEXT
90 GOTO 50
500 DATA21,237,62,20,178,62,21,237,62,20,178,62
510 DATA21,237,62,16,109,62,19,137,62,17,103,62
520 DATA14,162,125,0,0,250,8,180,62,10,247,62
530 DATA14,162,62,16,109,125,0,0,250
540 DATA10,247,62,13,208,62,16,109,62,17,103,125
550 DATA0,0,250,10,247,62,21,237,62,20,178,62
560 DATA21,237,62,20,178,62,21,237,62,16,109,62
570 DATA19,137,62,17,103,62,14,162,125,0,0,250
580 DATA8,180,62,10,247,62,14,162,62,16,109,125
590 DATA0,0,250,10,247,62,17,103,62,16,109,62
600 DATA14,162,62,0,0,125,14,162,125,0,0,250
610 DATA16,109,62,17,103,62,19,137,62,21,237,180
620 DATA13,10,62,23,59,62,21,237,62,19,137,180
630 DATA11,158,62,21,237,62,19,137,62,17,103,180
640 DATA10,247,62,19,137,62,17,103,62,16,109,125
650 DATA0,0,250,10,247,62,21,237,62,0,0,250
660 DATA0,0,250,21,237,62,43,219,62,0,0,250
670 DATA0,0,250,20,178,62,21,237,62,0,0,250
680 DATA0,0,250,20,178,62,21,237,62,19,137,62
690 DATA21,237,62,20,178,62,21,237,62,16,109,62
700 DATA19,137,62,17,103,62,14,162,125,0,0,250
710 DATA8,180,62,10,247,62,14,162,62,16,109,125
720 DATA0,0,250,10,247,62,13,208,62,16,109,62
730 DATA17,103,125,0,0,250,10,247,62,21,237,62
740 DATA20,178,62,21,237,62,20,178,62,21,237,62
750 DATA16,109,62,19,137,62,17,103,62,14,162,125
760 DATA0,0,250,8,180,62,10,247,62,14,162,62
770 DATA16,109,125,0,0,250,9,196,62,17,103,62
780 DATA16,109,62,14,162,125,0,0,250
790 DATA -1,-1,-1
800 POKE VO,0:POKE SI+4,0
```

Fig. 18 – Programma per suonare con il C64 il brano "Per Elisa".

Il programma in questione, che consente appunto di riprodurre il brano precedentemente codificato, è riportato nella *fig. 18*.

Tale programma inizia dalle istruzioni 3Ø e 35, le quali provvedono ad assegnare alle variabili SI e VO le locazioni di memoria del SID.

La linea di programma 4Ø immette (**POKE**) nella locazione di memoria SI+5 (54277) il valore 19Ø dell'attacco (A) e del decadimento (D).

Questo valore è ottenuto, come già precisato nella precedente lezione, dal seguente calcolo:

A=16*11=176, D=14 perciò A+D=19Ø

La stessa linea di programma 4Ø immette nella locazione di memoria SI+6 sostegno (S) e rilascio (R) il valore 216, ottenuto mediante il calcolo che segue:

S=16*13=208, R=8 perciò S+R=216

La linea di programma 45 provvede a regolare al valore massimo il volume sonoro.

La linea di programma 5Ø impone al computer di leggere, READ, le informazioni FH (byte alto), FL (byte basso), D (durata della nota), relative alle note da suonare riportate nelle istruzioni **DATA** (istruzioni 5ØØ÷79Ø).

All'istruzione 55 il programma si chiede se l'ultimo valore letto nelle istruzioni **DATA** è eguale a -1: in caso affermativo, l'elaborazione termina.

Le note musicali lette con l'istruzione 5Ø vengono inserite (POKE) nelle locazioni di memoria SI e SI+1 (istruzione 6Ø).

La linea di programma 65 definisce la forma d'onda triangolare (17) con la quale vengono riprodotte le note musicali.

Il ciclo di ritardo presente nella linea di programma 7Ø determina la durata del "sostenere".

L'istruzione 8Ø spegne l'onda triangolare (16=spento), ossia la annulla per il brevissimo tempo che impiega il computer a "contare" da 1 a 5Ø (istruzione 85).

L'istruzione 9Ø provvede, infine, a rimandare l'elaborazione alla linea di programma 5Ø.

La linea di programma 8ØØ azzera il volume e la forma d'onda triangolare.

Il programma ora descritto è valido per suonare qualsiasi brano musicale, modificando ovviamente soltanto le istruzioni DATA.

Terminato l'esame del programma, può immetterlo nel computer procedendo con molta attenzione, in quanto un errore commesso nel battere i dati relativi alle note può pregiudicare il funzionamento del programma stesso.

Un'ultima avvertenza molto importante è la seguente: occorre registrare il programma prima di eseguirlo con il comando RUN, poiché a causa di un errore commesso nell'immettere (POKE) il valore in una delle locazioni di memoria del SID il computer potrebbe bloccarsi e quindi non rispondere piú ai comandi, per cui sarebbe necessario spegnerlo perdendo cosí il programma appena battuto.

Pertanto, salvi dapprima per precauzione il programma, quindi lo esegua.

Dall'altoparlante del televisore verranno emesse le note del brano musicale "Per Elisa" di Beethoven.

Volendo ascoltare il brano in questione con una diversa velocità di esecuzione, dovrà modificare il valore finale dell'istruzione **FOR...**, presente nella linea di programma 85 (aumentando tale valore la velocità d'esecuzione diminuisce).

Inoltre, Lei potrà eventualmente variare i valori ADSR ed usare altre forme d'onda, al fine di ottenere suoni che simulino strumenti musicali oppure suoni totalmente nuovi.

Per esempio, per ottenere il suono simile a quello di uno xilofono, dovrà modificare la linea di programma 4Ø come segue:

```
4Ø POKE SI+5, 16*Ø+8:POKE SI+6, 16*Ø+8
```

Per ottenere il suono simile a quello di un violoncello dovrà modificare le istruzioni 4Ø, 65 e 8Ø come riportato di seguito:

```
40 POKE SI+5, 16*10:POKE SI+6, 16*10+3
   :
65 POKE SI+4,33      (onda a dente di sega)
   :
80 POKE SI+4,32      (spegne l'onda)
```

Per ottenere il suono di una tromba, dovrà apportare le seguenti modifi-

```
100 DATA 17,103,500,19,137,250,21,237,500,21,237,250
110 DATA 19,237,500,16,109,250,13,10,750
120 DATA 14,162,375,16,109,125,17,103,250,19,137,250
130 DATA 21,237,250,17,103,150,16,109,500,13,208,250
140 DATA 10,247,750,17,103,500,19,137,250,21,237,375
145 DATA 19,137,125,17,103,250
150 DATA 19,137,500,16,109,250,13,10,375
155 DATA 14,162,125,16,109,250
160 DATA 17,103,375,16,109,125,14,162,250,16,109,375
170 DATA 14,162,125,13,208,250,14,162,1500,0,0,1500
180 DATA 17,103,375,19,137,125,21,237,250
185 DATA 21,237,375,23,59,125,21,237,250
190 DATA 19,137,375,21,237,125,19,137,250
195 DATA 16,109,375,17,103,125,19,137,250
200 DATA 17,103,375,16,109,125,17,103,250
205 DATA 14,162,375,16,109,125,17,103,250
210 DATA 16,109,375,17,103,125,16,109,250
215 DATA 13,208,375,14,162,125,16,109,250
220 DATA 17,103,375,19,137,125,21,237,250
225 DATA 21,237,375,23,59,125,26,20,250
230 DATA 19,137,375,21,237,125,19,137,250
235 DATA 16,109,375,17,103,125,19,137,250
240 DATA 17,103,250,16,109,250,14,162,250
245 DATA 13,208,375,12,78,125,13,208,250
250 DATA 14,162,1500,0,0,1500
260 DATA -1,-1,-1
270 POKE VO,0:POKE SI+4,0
```

Fig. 19 – Codifica del brano musicale "Green Sleeves to a Ground".

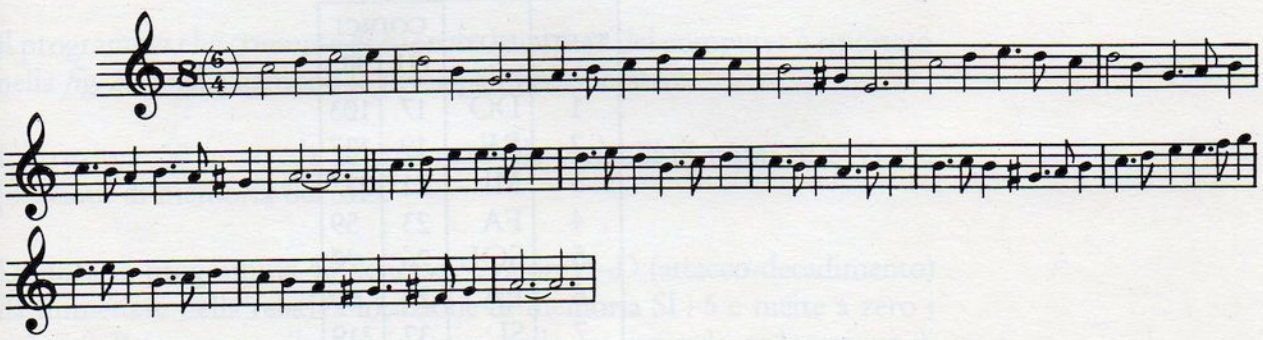


Fig. 20 – Spartito del brano musicale inglese "Green Sleeves to a Ground" (di autore anonimo).

che alle istruzioni 4Ø, 65, 8Ø:

```
4Ø POKE SI+5, 16*6:POKE SI+6,Ø
:
65 POKE SI+4,33
:
8Ø POKE SI+4,32
```

Nella *fig. 19* è riportata la codifica di un antico brano musicale inglese, denominato "Green Sleeves to a Ground", il cui spartito è rappresentato nella *fig. 20*; Lei potrà riprodurre tale brano utilizzando il programma della *fig. 18*, di cui ovviamente dovrà sostituire le istruzioni DATA da 5ØØ a 8ØØ con quelle relative al nuovo brano musicale.

5.1 – Una tastiera musicale con il C64

Eseguiamo ora un interessante esercizio di programmazione, che consente di suonare la tastiera del computer come se fosse quella di un pianoforte.

Per ottenere ciò occorre, in primo luogo, fare corrispondere le note musicali, necessarie per riprodurre la melodia prescelta, ad alcuni tasti del C64.

Nell'esempio che prenderemo in considerazione saranno utilizzate otto note musicali della scala DO-maggiore, le quali verranno fatte corrispondere ai tasti numerati da 1 a 8, come indicato nella tabella riportata di seguito, nella quale sono anche elencati i codici relativi alle note usate.

TASTI	NOTE	CODICI	
		FH	FL
1	DO	17	1Ø3
2	RE	19	137
3	MI	21	237
4	FA	23	59
5	SOL	26	2Ø
6	LA	29	69
7	SI	32	219
8	DO	34	2Ø7

```
10 REM*PIANOFORTE SCALA MUSICALE IN DO*
15 PRINT"J"
20 PRINT TAB(9) "■■SUONA CON LA TASTIERA"
30 SI=54272:VO=54296
40 POKE SI+5,9 :POKE SI+6,0
45 POKE SI+2,0:POKE SI+3,9:POKE VO,15
50 REM*ARCHIVIA SCALA IN DO*
55 FOR N=1 TO 8
60 READ FH(N),FL(N)
65 NEXT N
80 REM*SUONA CON LA TASTIERA*
85 GET A$:IF A$=""THEN 85
90 N=VAL(A$)
93 POKE SI+4,0
95 IF  N=0 OR N=9 THEN 175
100 PRINT
105 IF N=1 THEN PRINT"DO"
110 IF N=2 THEN PRINT"RE"
120 IF N=3 THEN PRINT"MI"
125 IF N=4 THEN PRINT"FA"
130 IF N=5 THEN PRINT"SOL"
135 IF N=6 THEN PRINT"LA"
140 IF N=7 THEN PRINT"SI"
145 IF N=8 THEN PRINT"DO"
150 FOR T=1 TO 20:NEXT
155 POKE SI,FL(N)
160 POKE SI+1,FH(N)
162 POKE SI+4,65
165 GOTO 85
175 END
500 DATA 17,103,19,137,21,237,23,59
510 DATA 26,20,29,69,32,219,34,207
```

Fig. 21 – Programma BASIC per suonare la tastiera del C64 come quella di un pianoforte.

Il programma che consente di suonare la tastiera del computer è riportato nella *fig. 21*; analizziamone le principali fasi operative.

L'istruzione 3Ø provvede ad assegnare alle variabili di nome SI e VO le locazioni di memoria del SID.

La linea di programma 4Ø definisce i valori A–D (attacco-decadimento) da immettere nella relativa locazione di memoria SI+5 e mette a zero i valori S–R (sostegno-rilascio) immessi nella corrispondente locazione di memoria SI+6.

La linea di programma 45 mette a zero il valore dell'ampiezza del byte basso, relativo alla forma d'onda quadra (SI+2) ed assegna il valore 9 all'ampiezza del byte alto della stessa forma d'onda quadra (SI+3). Infine, tale linea di programma imposta il volume sonoro al massimo (15).

Le istruzioni 55÷65 servono a memorizzare nelle matrici FH(N) e FL(N) ad un indice (vettori) i codici delle note musicali archiviate nelle istruzioni **DATA** (linee di programma 5ØØ-51Ø).

L'istruzione 85 consente l'immissione da tastiera di uno degli otto tasti numerici: **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**.

Battendo uno di questi tasti, dall'altoparlante del televisore verrà emessa la corrispondente nota musicale della scala DO-maggiore.

In particolare, l'istruzione 9Ø, mediante la funzione stringa **VAL** (A$), calcola il valore numerico corrispondente al tasto che è stato pigiato ottenendo cosí un numero compreso fra 1 e 8.

Questo numero viene quindi usato come indice per prelevare dai vettori FH(N) e FL(N) il codice della nota corrispondente.

Se si batte il tasto relativo al numero 1, si attiva la nota DO (codici 17 e 1Ø3); se si batte il tasto 2, si attiva la nota RE (codici 19 e 137) e cosí via.

Se invece dei tasti numerati da 1 a 8 si batte il tasto Ø oppure il tasto 9 (istruzione 95), o qualsiasi altro tasto, l'elaborazione salta all'istruzione 175, dove termina il programma.

L'istruzione 93 provvede ad azzerare la locazione di memoria relativa alla forma d'onda (SI+4) prima di suonare la nota desiderata.

Le linee di programma comprese fra 1Ø5 e 145 provvedono a visualizzare sullo schermo il nome relativo alla nota suonata, mentre l'istruzione 15Ø crea un breve intervallo di silenzio fra una nota e l'altra.

Le istruzioni 155 e 16Ø provvedono a suonare la nota voluta.

Infine, la linea di programma 162 attiva la forma d'onda quadra (65), con la quale vengono appunto emesse le note musicali.

Batta ora sulla tastiera il programma descritto e lo esegua con il comando **RUN**.

Sullo schermo comparirà la scritta:

SUONA CON LA TASTIERA

A questo punto può provare a suonare un brano musicale qualsiasi, oppure quello riportato nella *fig. 22*, pigiando nell'ordine i tasti numerici della tastiera elencati in tale figura.

```
1 3 5 5 5 6 5 3 1 2 3 3 2 1 2
1 3 5 5 5 6 5 3 1 2 3 3 2 2 1
4 4 6 6 6 5 5 3 1 2
1 3 5 5 5 6 5 3 1
2 3 3 2 2 1
```

Fig. 22 – Brano musicale da provare a suonare sulla tastiera del C64.

Dall'altoparlante del televisore verranno riprodotte le note del brano musicale suonate e il loro nome sarà visualizzato sullo schermo.

Al termine dell'esecuzione batta il tasto **0** oppure il tasto **9**.

Per fare in modo che la tastiera del computer possa essere usata per suonare un altro brano musicale, occorre rieseguire il programma con il comando **RUN**.

Il programma eseguito può essere ampliato e perfezionato al fine di poter suonare, oltre alle otto note previste, altre note musicali, facendole corrispondere ad altri tasti del computer.

5.2 – Effetti sonori con il C64

Con il C64 si possono creare effetti sonori di vario tipo, che vengono utilizzati per migliorare e ravvivare i programmi.

A questo scopo il computer dispone di una particolare forma d'onda, contraddistinta dal numero 129, la quale, quando è inserita nella locazione di memoria (SI+4), relativa appunto a tale forma d'onda, produce un rumore simile ad un fruscio (denominato **rumore bianco**), paragonabile a quello emesso da un televisore acceso quando non riceve alcuna emittente.

Con la forma d'onda del rumore possono essere utilizzate le diverse note musicali allo scopo di creare vari tipi di rumore bianco: ad esempio, per

simulare esplosioni, spari, onde del mare, ecc.

Per creare effetti sonori si possono ovviamente usare anche le forme d'onda triangolare, a dente di sega e quadra.

Non vi sono regole precise da seguire per ottenere gli effetti desiderati, per cui si deve procedere per tentativi sino al raggiungimento del risultato voluto.

In linea di massima, è bene seguire i seguenti suggerimenti: alternare rapidamente il suono di due note per creare effetti d'eco; variare i parametri ADSR della forma d'onda per alterare le caratteristiche di un suono; utilizzare contemporaneamente diverse forme d'onda per ottenere effetti speciali; ecc.

Gli effetti sonori possono essere incorporati nel programma mediante strutture di diramazione semplice tipo **IF... THEN...**, tramite alternative multiple **ON... GOTO...** e, nei casi piú complessi, per mezzo di subroutine.

A titolo esemplificativo nella *fig. 23* sono riportati quattro tipici programmi di effetti sonori.

Il programma "ESPLOSIONE" utilizza il generatore di rumore (SI+4, 129), linea di programma 5Ø, per produrre appunto l'effetto sonoro di una esplosione.

Le istruzioni presenti nelle linee di programma 3Ø e 4Ø definiscono i valori ADSR dell'onda ed i parametri della nota DO (103 e 17).

Il ciclo di ritardo presente alla linea di programma 6Ø determina la durata dell'esplosione.

Nel programma "SPARO" l'istruzione 3Ø inizia un ciclo **FOR...** che termina all'istruzione 7Ø; tale ciclo ha la funzione di fare diminuire il volume sonoro, portandolo dal valore massimo (15) al valore zero della forma d'onda del rumore (linea di programma 6Ø).

Nel programma "ONDE DEL MARE" è ancora usata la forma d'onda del rumore (istruzione 5Ø) per simulare appunto l'effetto delle onde che si infrangono sulla riva.

Il ciclo di ritardo della linea di programma 6Ø determina la durata della pausa fra un'ondata e l'altra.

```
10 REM*ESPLOSIONE*
20 SI=54272:VO=54296
30 POKE SI+5,12:POKE SI+6,15
40 POKE SI,103:POKE SI+1,17
50 POKE VO,15:POKE SI+4,129
60 FOR T=1 TO 2500:NEXT T
70 POKE SI+4,0:POKE VO,0
```

```
10 REM*SPARO*
20 SI=54272:VO=54296
30 FOR X=15 TO 0 STEP -1
40 POKE VO,X :POKE SI+5,15
50 POKE SI+6,0:POKE SI+1,40
60 POKE SI,200:POKE SI+4,129
70 NEXT X
80 POKE SI+4,0:POKE SI+5,0
```

```
10 REM*ONDE DEL MARE*
20 SI=54272:VO=54296
30 POKE SI,0:POKE SI+1,255
40 POKE SI+5,219:POKE VO,15
50 POKE SI+4,129
60 FOR T=1 TO 5000:NEXT
70 POKE SI+4,0
80 GOTO 30
```

```
10 REM*RUMORE PER GIOCHI*
20 SI=54272:VO=54296
30 POKEVO,15
70 POKE SI+5,15:POKE SI+4,17
90 POKE SI+6,15
100 FOR X=1 TO 250 STEP 1:POKE SI+1,X:POKE SI+1,255-X:NEXT
120 POKE SI+4,16:POKE VO,0
130 GOTO 30
```

Fig. 23 – Programmi per ottenere effetti sonori con il C64.

Infine, il programma "RUMORE PER GIOCHI" crea una serie di suoni che possono servire per accompagnare il movimento di navi spaziali, alieni che compaiono sullo schermo.

Le linee di programma 7Ø–9Ø definiscono, come al solito, i valori ADSR dell'onda triangolare (SI+4,17).

L'istruzione **FOR...**, presente nella linea di programma 1ØØ, provvede a variare ciclicamente il valore della variabile X da 1 a 250.

Tale valore viene immesso nella locazione di memoria SI+1, che controlla il byte alto della frequenza della nota generata.

Contemporaneamente lo stesso valore di X viene sottratto dal massimo valore (255) del byte alto della frequenza, ottenendo cosí un particolare effetto di vibrato.

Scriva ora ed esegua i quattro programmi esaminati, onde verificarne praticamente gli effetti prodotti.

Ricordi di cancellare la memoria del computer prima di immettere in esso ciascun programma.

6. – Suoni ed effetti sonori con il VIC-20

Nella precedente lezione ha avuto modo di apprendere come si possa programmare un brano musicale con il computer.

Vediamo ora un interessante programma che consente invece di suonare la tastiera del computer come se fosse quella di un pianoforte.

Per ottenere ciò occorre fare corrispondere le note musicali, necessarie per riprodurre il brano musicale prescelto, ad alcuni tasti del VIC-20.

Nell'esempio che prenderemo in considerazione saranno utilizzate otto note musicali della scala SI–maggiore, le quali verranno fatte corrispondere ai tasti numerati da 1 a 8, come indicato nella tabellina riportata di seguito, nella quale sono anche elencati i codici relativi alle note usate.

TASTI	1	2	3	4	5	6	7	8
NOTE	SI	DO #	RE #	MI	FA #	SOL #	LA #	SI
CODICI	223	227	230	231	234	236	238	239

Il programma che consente di suonare la tastiera del computer è riportato nella *fig. 24*; analizziamone le principali fasi operative.

```
10 REM*  PIANOFORTE  *
15 REM*SCALA MUSICALE*
20 REM*  IN SI MAG.  *
25 PRINT"SUONA CON LA TASTIERA"
30 S2=36875
35 V=36878
40 POKE S2,0
45 REM*ARCHIVIA SCALA*
50 REM* SI MAGGIORE  *
60 FOR N=1 TO 8
70 READ A(N)
80 NEXT N
85 DATA 223,227,230
90 DATA 231,234,236
100 DATA 238,239
150 REM*SUONA LA*
160 REM*TASTIERA*
200 POKE V,3
220 GET A$:IF A$=""THEN 220
230 N=VAL(A$)
240 IF N=0 OR N=9 THEN 310
242 PRINT
245 IF N=1 THEN PRINT"SI"
250 IF N=2 THEN PRINT"DO#"
255 IF N=3 THEN PRINT"RE#"
260 IF N=4 THEN PRINT"MI"
265 IF N=5 THEN PRINT"FA#"
270 IF N=6 THEN PRINT"SOL#"
275 IF N=7 THEN PRINT"LA#"
280 IF N=8 THEN PRINT"SI"
285 POKE S2,0
290 FOR T=1 TO 25:NEXT T
295 POKE S2,A(N)
300 GOTO 220
310 POKE S2,0
320 END
```

Fig. 24 – Programma BASIC per suonare la tastiera del VIC-20 come quella di un pianoforte.

Le istruzioni 3Ø÷4Ø provvedono ad assegnare alle variabili di nome S2 e V le locazioni di memoria riservate rispettivamente alla tonalità media (TONO2) ed al volume: in questo modo verranno usati nel programma i nomi suddetti al posto degli indirizzi 36875 e 36878.

In particolare, l'istruzione 4Ø pone a zero la tonalità media (S2) prima di eseguire il programma.

Le istruzioni da 6Ø a 8Ø servono a memorizzare nella matrice A(N) ad un indice (vettore) i codici delle note musicali archiviate nelle istruzioni **DATA** (linee di programma 85÷1ØØ).

L'istruzione 2ØØ imposta il volume sonoro al valore 3 (come già precedentemente precisato, il volume sonoro è indicato da un valore intero compreso fra Ø e 15).

L'istruzione 22Ø consente l'immissione da tastiera di uno degli otto tasti: **1, 2, 3, 4, 5, 6, 7, 8.**

Battendo uno di questi tasti numerici, dall'altoparlante del televisore verrà emessa la corrispondente nota musicale della scala SI-maggiore.

Questa nota durerà fino a che non verrà battuto un altro tasto (cioè suonata un'altra nota).

In particolare, l'istruzione 23Ø calcola, mediante la funzione stringa **VAL** (A$), il valore corrispondente al tasto che è stato pigiato, ottenendo così un numero compreso fra 1 e 8.

Questo numero viene quindi usato come indice per prelevare dalla matrice A(N) il codice della nota corrispondente.

Se si batte il tasto relativo al numero 1, viene attivata la nota SI (codice 223); se si batte il tasto 2, si attiva la nota DO # (codice 227) e così via.

La linea di programma 295 provvede appunto a fare suonare la nota voluta.

Se invece dei tasti numerati da 1 a 8 si batte il tasto **Ø** oppure il tasto **9** (istruzione 24Ø), o qualsiasi altro tasto, l'elaborazione salta all'istruzione 31Ø, che pone a zero il volume sonoro e quindi il programma ha termine.

Le linee di programma comprese fra 245 e 28Ø provvedono a visualizzare sullo schermo il nome relativo alla nota suonata.

Infine, l'istruzione 29Ø crea un breve intervallo di silenzio fra le note.

Immetta ora nel computer il programma descritto e lo esegua con il comando **RUN**.

Sullo schermo comparirà la scritta:

SUONA CON LA TASTIERA

A questo punto può provare a suonare un brano musicale qualsiasi, oppure quello riportato nella *fig. 25*, pigiando nell'ordine i tasti numerici della tastiera elencati in tale figura.

1 3 5 5 5 6 5 3 1 2 3 3 2 1 2
1 3 5 5 5 6 5 3 1 2 3 3 2 2 1
4 4 6 6 6 5 5 3 1 2
1 3 5 5 5 6 5 3 1
2 3 3 2 2 1

Fig. 25 – Brano musicale da provare a suonare sulla tastiera del VIC-20.

La durata di ciascuna nota dipenderà ovviamente dalla velocità con la quale viene eseguito il brano.

Dall'altoparlante del televisore verranno riprodotte le note del brano musicale suonate, e il loro nome sarà visualizzato sullo schermo.

Al termine dell'esecuzione batta il tasto **Ø** oppure il tasto **9** per interrompere l'ultima nota suonata.

Per fare in modo che la tastiera del computer possa essere usata per suonare un altro brano musicale, occorre rieseguire il programma con il comando **RUN**.

Il programma eseguito può essere ampliato e perfezionato al fine di poter suonare, oltre alle otto note previste, altre note musicali, facendole corrispondere ad altri tasti del computer.

6.1 – Effetti sonori con il VIC-20

Con il computer VIC-20 si possono creare effetti sonori di vario tipo, che vengono utilizzati per migliorare e ravvivare i programmi.

A questo scopo il computer dispone di una particolare locazione di memoria, contraddistinta con il valore 36877 (voce o tono 4), la quale, come le altre tre voci, dispone di tre ottave.

Tale locazione di memoria produce un fruscio (denominato **rumore bianco**), simile a quello emesso da un televisore acceso quando non riceve alcuna emittente.

Provi, per esempio, a battere le seguenti istruzioni BASIC ad esecuzione immediata:

```
POKE 36878,15
POKE 36877,135
```

locazione di memoria relativa al rumore — nota DO

Dopo aver regolato il volume del televisore, udrà un suono simile (con un po' di fantasia) a quello prodotto da un aereo ad elica.

Per interrompere il suono batta l'istruzione:

```
POKE 36877,Ø
```

A titolo esemplificativo, nella *fig. 26* sono riportati tre tipici programmi di effetti sonori.

Il programma "ESPLOSIONE" utilizza il generatore di rumore per produrre appunto l'effetto sonoro di una esplosione.

Il ciclo di programma, che inizia all'istruzione 2Ø, ha il compito di fare diminuire il valore del volume sonoro, portandolo dal massimo (1Ø) a zero, in modo che il rumore prodotto dall'esplosione diminuisca e si annulli naturalmente.

Il ciclo di ritardo presente nella linea di programma 3Ø determina la durata dell'esplosione.

```
10 REM*ESPLOSIONE*
15 POKE 36877,135
20 FOR T=10 TO 0 STEP -1
25 POKE 36878,T
30 FOR A=1 TO 200:NEXT A
35 NEXT T
40 POKE 36877,0
45 POKE 36878,0
50 END
```

```
10 REM*SIRENA BITONALE*
15 V=36878
20 S2=36875
25 S3=36876
30 POKE V,15
35 FOR D=1 TO 10
40 POKE S2,212
45 FOR T=1 TO 500:NEXT T
50 POKE S2,0
55 POKE S3,212
60 FOR T=1 TO 500:NEXT T
65 POKE S3,0
70 NEXT D
75 POKE V,0
80 END
```

```
10 REM*SUONO DEL TELEFONO"
15 V=36878
20 S3=36876
22 POKE V,15
25 FOR A=1 TO 5
30 FOR B=1 TO 50
35 POKE S3,219
40 FOR N=1 TO 9:NEXT N
45 POKE S3,0
50 NEXT B
55 FOR T=1 TO 2000:NEXT
60 NEXT A
65 POKE V,0
70 END
```

Fig. 26 - Programmi per ottenere effetti sonori con il VIC-20.

Nel programma "SIRENA BITONALE" sono utilizzate, invece, due tonalità: quella media (locazione 36875) e quella alta (locazione 36876), entrambe con codice nota 212 (FA #).

Il ciclo di elaborazione che inizia alla linea di programma 35 e termina alla linea 7Ø provvede a fare suonare dieci volte prima la nota con tonalità media e poi la nota con tonalità piú alta.

La durata di ciascuna tonalità è determinata dai due cicli di ritardo presenti nelle linee di programma 45 e 6Ø.

Infine, nel programma "SUONO DEL TELEFONO" l'istruzione 25 inizia un ciclo di elaborazione che determina cinque trilli del telefono.

Il secondo ciclo presente nell'istruzione 3Ø provvede a stabilire la durata di ciascun trillo, mentre l'istruzione 35 fa suonare la nota LA (codice 219) con tonalità alta (locazione di memoria 36876).

Il ciclo di ritardo della linea di programma 4Ø determina brevi interruzioni durante il suono delle note, creando cosí un effetto di vibrato che simula appunto il trillo del telefono.

Il secondo ciclo di ritardo, presente alla linea di programma 55, stabilisce la durata della pausa di silenzio fra un trillo e l'altro.

Scriva ora ed esegua i tre programmi esaminati, allo scopo di verificarne praticamente gli effetti prodotti.

Ricordi di cancellare la memoria del computer prima di immettere in esso ciascun programma.

Come ha potuto constatare, utilizzando le apposite locazioni di memoria del computer relative al controllo dei suoni è possibile creare svariati effetti sonori.

Non vi sono regole precise da seguire per ottenere gli effetti desiderati, per cui occorre procedere per tentativi sino ad ottenere il risultato voluto.

* * * * * * *

In seguito...

Nella prossima lezione verrà esaminato il sistema di utilizzazione dei file-dati, ossia degli archivi d'informazione, con il registratore magnetico.

CONTROLLO DIDATTICO 11

Domande

1) - Che cosa sono ed a che cosa servono le funzioni stringa?

.....

.....

.....

.....

.....

2) - Quali sono le funzioni stringa che consentono la visualizzazione dei codici ASCII dei caratteri simbolici riportati sulla tastiera del computer?

.....

.....

.....

.....

3) - Quali funzioni stringa consentono di accedere a singoli o a gruppi di caratteri di una stringa?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

4) - Se si scrive l'istruzione:

```
PRINT VAL ("100A")
```

che cosa apparirà sullo schermo?

.....

5) - Se si scrivono le istruzioni:

```
X=3.14159
PRINT STR$ (X)
```

che cosa verrà visualizzato sullo schermo?

.....

.....

6) - Descrivere le funzioni operative svolte dal seguente, semplice programma:

```
10 A$="VENTIMIGLIA"
20 PRINT LEFT$(A$,5)
30 PRINT MID$(A$,6,2)
40 PRINT RIGHT$(A$,4)
50 END
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

7) - Utilizzando solamente le funzioni **MID$**, scrivere un breve programma che consenta di ottenere dalla stringa A$="SALTIMBANCO" due sottostringhe: una contenente i caratteri SALTI e l'altra i caratteri BANCO.

.....

.....

.....

.....

.....

.....

.....

8) - Utilizzando le funzioni **CHR$** e **LEFT$**, scrivere un programma che visualizzi sullo schermo quattro palline (codice 113) di colore rosso, verde, giallo, azzurro.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

9) - Scrivere un programma che analizzi qualsiasi stringa immessa nel computer ed indichi se essa è costituita da piú di 1Ø caratteri oppure da meno di 1Ø caratteri oppure esattamente da 1Ø caratteri.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

10) - A che cosa serve la funzione di tempo **TI$**?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

11) - Scrivere un programma che individui se in una serie di parole (stringhe) immesse nel computer è presente la parola "BASIC" e quindi la visualizzi sullo schermo. Per immettere le stringhe si dovrà prevedere un vettore A$ a 50 elementi.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

12) - Tracciare il flow-chart del programma contemplato dalla domanda n. 11.

Risposte

1) - Le funzioni stringa consentono di operare su una stringa alfanumerica o su parte di essa. Tutte le funzioni stringa sono costituite da una parola mnemonica (in lingua inglese), seguita da un argomento racchiuso fra due parentesi tonde.

2) - La visualizzazione dei codici ASCII dei caratteri riportati sulla tastiera è consentita dalle funzioni stringa **ASC** e **CHR$**.

3) - Le funzioni stringa che permettono di accedere a singole parti di una stringa sono le seguenti: **LEN**, **RIGHT$**, **LEFT$** e **MID$**.
La funzione **LEN** calcola il numero dei caratteri di una stringa; le funzioni **RIGHT$** e **LEFT$** individuano rispettivamente i caratteri piú a destra e quelli piú a sinistra; la funzione **MID$** individua gruppi o singoli caratteri disposti all'interno di una stringa.

4) - Scrivendo tale istruzione, sullo schermo dovrebbe vedersi il numero 1ØØ.

5) - Con le istruzioni citate sullo schermo apparirà il numero 3.14159, cosí come è stato assegnato alla variabile numerica di nome X.

6) - Nell'istruzione 1Ø del programma presentato viene assegnata alla variabile di nome A$ la stringa "VENTIMIGLIA". La linea di programma 2Ø fornisce una sottostringa costituita dai primi cinque caratteri (VENTI), a partire da sinistra, della stringa A$.
La linea di programma 3Ø provvede a fornire una sottostringa di due caratteri (MI), a partire dal carattere 6, della stringa A$. Infine, la linea di programma 4Ø estrae una sottostringa di quattro caratteri (GLIA), a partire da destra, della stringa A$.

7) - Un possibile listato di tale programma è riportato di seguito:

```
10 A$="SALTIMBANCO"
20 PRINT MID$(A$,1,5)
30 PRINT MID$(A$,7,5)
40 END
```

Come si può notare, con **MID$** si possono ottenere anche le funzioni **LEFT$** e **RIGHT$**.

8) - Il programma richiesto può avere il seguente aspetto:

```
10 PRINT CHR$(147)
20 C$=CHR$(28)+CHR$(30)+CHR$(158)+CHR$(159)
30 FOR K=1 TO 4
40 PRINT LEFT$(C$,K);CHR$(113);
50 NEXT
60 PRINT CHR$(17)
70 END
```

9) - Una possibile codifica in BASIC di tale programma è la seguente:

```
10 PRINT"□"
15 PRINT"INTRODURRE LA STRINGA"
20 INPUT A$
30 IF LEN(A$)<10 THEN 65
40 IF LEN(A$)=10 THEN 90
50 PRINT"E'MAGGIORE DI 10 CARATTERI"
60 GOTO 100
65 PRINT"E'MINORE DI 10 CARATTERI"
80 GOTO 100
90 PRINT"E'DI 10 CARATTERI"
100 END
```

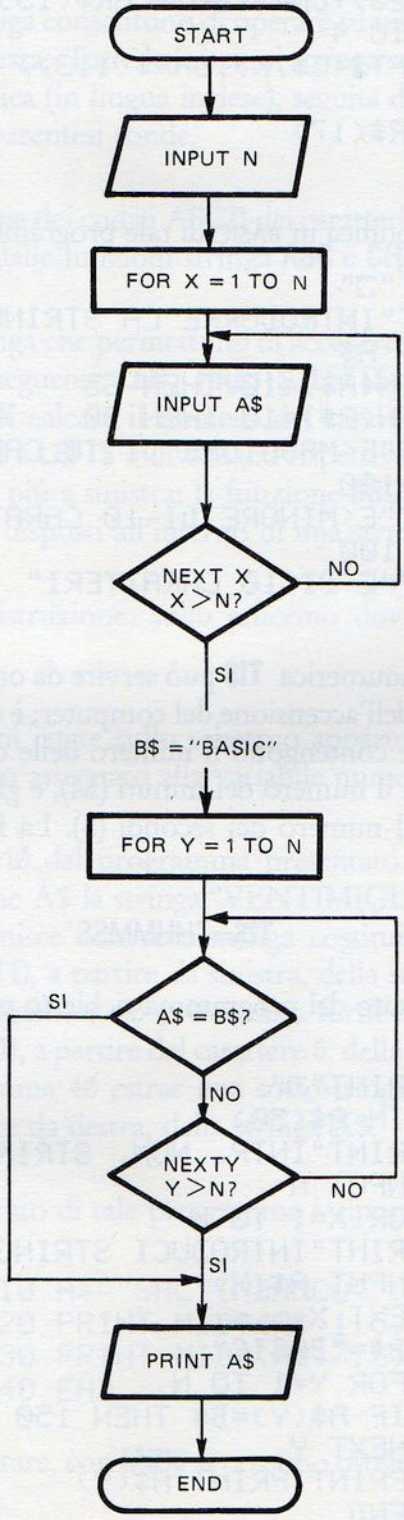
10) - La funzione alfanumerica **TI$** può servire da orologio e viene attivata al momento dell'accensione del computer; è costituita da sei caratteri: i primi due contengono il numero delle ore (H), il terzo ed il quarto carattere il numero dei minuti (M), e gli ultimi due caratteri rappresentano il numero dei secondi (S). La forma della funzione TI$ è quindi:

TI$="HHMMSS"

11) - Un possibile listato del programma richiesto può essere il seguente:

```
10 PRINT"□"
15 DIM A$(50)
20 PRINT"INTR. NUM. STRINGHE"
30 INPUT N
40 FOR X=1 TO N
50 PRINT"INTRODUCI STRINGA"
60 INPUT A$(X)
90 NEXT X
100 B$="BASIC"
110 FOR Y=1 TO N
130 IF A$(Y)=B$ THEN 150
140 NEXT Y
150 PRINT:PRINT A$(Y)
160 END
```

12) - Il flow-chart del programma relativo alla domanda n. 11 può avere il seguente aspetto:



PARLA BASIC

Corso teorico-pratico per corrispondenza di

ANALISI E PROGRAMMAZIONE BASIC

Scuola Radio Elettra

10126 Torino - Via Stellone 5

Tel. (011) 674432